ATARI ST ATARI TT







NVDI

2.10

NVDI
2.10

Programmautoren: Sven & Wilfried Behne
Handbuchautoren: Sven & Wilfried Behne

Satz: BELA Computer GmbH
Umschlaggestaltung: BELA Computer GmbH

Inhaltsverzeichnis

Einleitung

„Hören Sie auf, mich zu verwirren!“ (Sledge Hammer)

Computer scheinen nicht nur immer schneller, sondern auch immer komplexer und undurchschaubarer zu werden. Diese Anleitung soll Ihnen helfen, NVDI effektiv einzusetzen und an Ihre persönliche Konfiguration anzupassen.

Durch die optimierten Treiber und das GDOS-Konzept von NVDI steht Ihnen jetzt ein System zur Verfügung, das eine einheitliche Ansteuerung der Peripheriegeräte gestattet, eine sehr hohe Kompatibilität zum ATARI-VDI aufweist und bei allen Bildschirmausgaben über das Betriebssystem zu einer deutlichen Geschwindigkeitssteigerung führt.

NVDI 2.10 bietet:

ein (abschaltbares) GDOS

stark beschleunigte VDI-Funktionen

Unterstützung und Beschleunigung aller ST/TT-Monochrom- und Farbmodi

neue, extrem leistungsfähige Textroutine

Anpassung an die neueren Prozessoren der M680xx-Serie (M68010/20/30). NVDI enthält optimierte Spezialroutinen für M68020/68030.

nachladbare Systemzeichensätze

Auflösungsunabhängigkeit. NVDI kommt auch mit anderen Bildschirmformaten als der Standardauflösung (640 * 400) klar. NVDI wurde erfolgreich unter BIGSCREEN, der Grafikkarte MegaScreen, und der Grafikerweiterung AutoSwitch-OverScan getestet.

Beschleunigung der BIOS- und GEMDOS-Zeichenausgabe, mit problemloser GEMDOS-Ausgabeumlenkung

GEM2.x-Kompatibilität

GEM/3-Bezierfunktionen

ausschaltbare Line-A-Funktionen

läuft zusammen mit FSM-GDOS

extraschnelle Routinen für den ATARI TT

MultiTOS-Anpassung

Systemvoraussetzungen

Der Betrieb des NVDI ist bereits mit 512kB Speicher möglich. Bei Benutzung vieler Fonts und/oder Gerätetreiber ist jedoch eine Festplatte oder mindestens 1Mb Speicher empfehlenswert.

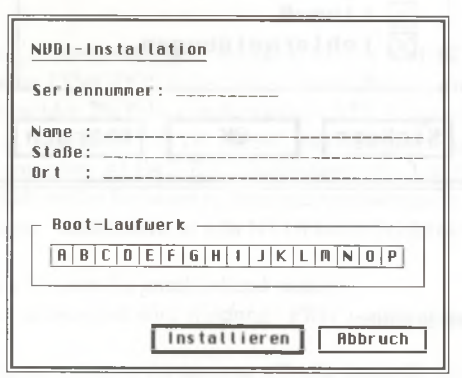
Da auf die Verwendung selbstmodifizierenden Codes verzichtet wurde und die Veränderungen der M680xx-Serie seit dem M68010 berücksichtigt würden, arbeitet das NVDI problemlos mit neueren Prozessoren von Motorola.

Platzbedarf auf Massenspeicher: ca. 160 kB
minimaler Speicherbedarf: ca. 80 kB
maximale Anzahl von Gerätetreibern: 99
maximale Anzahl von Fonts: nur vom Hauptspeicher abhängig
maximale Anzahl Handles: 128

Installation

Gehen Sie bitte wie folgt vor:

Legen Sie die Originaldiskette in Laufwerk A.
Starten Sie das im Hauptverzeichnis liegende INSTALL.PRG. Danach erscheint die folgende Dialogbox:



Geben Sie die Seriennummer ein, die auf der Registrierkarte und dem Diskettenetikett aufgedruckt ist. Seien Sie dabei sehr sorgfältig, die Nummer muß exakt stimmen. Anschließend geben Sie Ihren Namen und Ihre Adresse ein.
Selektieren Sie das Startlaufwerk Ihres Systems (bei Diskettenbetrieb normalerweise Laufwerk A, bei Festplatten oftmals Partition C).
Klicken Sie auf den Button „Installieren“.
Wenn Sie NVDI auf dem Laufwerk installieren möchten, in dem die Originaldiskette liegt, so werden

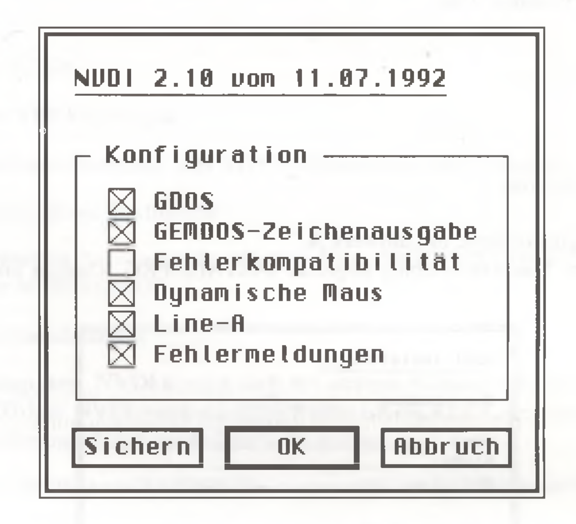
Sie aufgefordert, die Diskette zu wechseln.
Eine Alert-Box signalisiert Ihnen das Ende der Installation.
Drücken Sie auf den Reset-Knopf. Das NVDI wird dann während des Systemstarts geladen, aktiviert und verbleibt resident im Speicher.

Durch Drücken beider Shift-Tasten kann NVDI während des Systemstarts deaktiviert werden.

Beachten Sie bitte auch die Datei README.1ST, die aktuelle Informationen enthält.

NVDICONF - NVDI-Konfiguration

Über dieses Accessory läßt sich das NVDI während des Betriebs konfigurieren. Beim Aufruf des Accessories erscheint die folgende Dialogbox:



Die Bedeutung der einzelnen Schalter:

GDOS

Schaltet GDOS ein/aus. Das Ausschalten ist immer dann erforderlich, wenn Programme nicht korrekt mit GDOS zusammenarbeiten, d.h. unsauber sind.

GEMDOS-Zeichenausgabe

Ist dieser Schalter aktiviert, so wird die GEMDOS-Zeichenausgabe nochmals beschleunigt. Im Gegensatz zu den üblichen Lösungen funktioniert die Zeichenausgabe auch mit Ausgabeumlenkung.

Fehlerkompatibilität

Ist dieser Schalter aktiviert, so werden verschiedene fehlerhafte VDI-Aufrufe/Funktionen konform zum ATARI-VDI behandelt. Näheres s. Abschnitt „Fehlerkompatibilität".

Dynamische Maus

NVDI enthält eine dynamische Mausroutine. Diese reagiert bei langsamen Mausbewegungen linear, bei schnelleren Mausbewegungen progressiv. Die Benutzung dieser dynamischen Mausroutine bietet sich besonders bei Benutzung eines Großbildschirms an.

Line-A

Dieser Knopf hat eine besondere Bedeutung: Wenn Ihr Programm nach Deselektierung dieses Knopfes problemlos läuft, so können Sie davon ausgehen, daß keine der sogenannten LINE-A-Funktionen von ihm benutzt werden, was i.a. eine Grundvoraussetzung für die problemlose Lauffähigkeit dieses Programmes auf Großbildschirmen ist.
Da auf dem ATARI ST/TT noch viele Programme zu finden sind, die diese Funktionen benutzen, sollten Sie diesen Knopf standardmäßig selektieren (dies ist auch die NVDI-Voreinstellung).

Achtung: Bei der Zusammenarbeit mit anderen GDOSsen sind einige der Schalter nicht anwählbar.
Ist ein anderes GDOS installiert, dann wird das NVDI-GDOS ausgeschaltet. Ist FSM-GDOS installiert, so wird zusätzlich der Line-A-Schalter aktiviert.
Wenn statt des NVDI-Bildschirmtreibers ein anderer Bildschirmtreiber benutzt wird, so wird das NVDI-GDOS aktiviert, und alle anderen Schalter werden gesperrt.

Fehlermeldungen

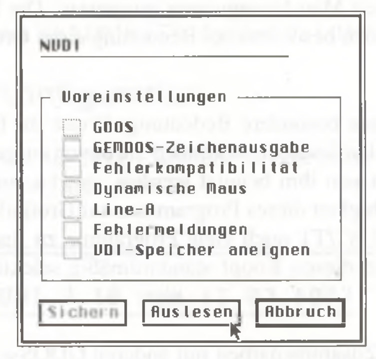
Ist der Schalter „Fehlermeldungen" aktiviert, so gibt NVDI Fehlermeldungen in ALERT-Boxen (bzw. unter FSMGDOS in der linken oberen Bildschirmecke) aus. Um Abstürze mit Software zu vermeiden, die Fehler im Bereich der AES-Programmierung aufweist [fehlendes *wind_update()*], verzichtet NVDI darauf, bei fehlgeschlagenem *v_opnvwk()* auf die Fehlerursache mit einer ALERT-Box hinzuweisen (die von NVDI ausgegebene ALERT-Box kann bei fehlerhafter Software zu Reentranzproblemen im AES führen). Ist der Schalter „Fehlermeldungen" deaktiviert, so gibt NVDI keine Fehlermeldungen mehr aus.

Die gewünschte Einstellung können Sie anschließend sichern.
NVDICONF läuft, wenn es umbenannt wird (Endung '.PRG'), auch als normales GEM-Programm.

Im Ordner CPX finden Sie ein CPX-Modul für das neue ATARI-Kontroll-Accessory (XCONTROL.ACC), welches dieselben Funktionen wie das NVDICONF-Accessory bietet.

NVDIDFLT - NVDI-Standardeinstellungen

Wenn Sie einen Accessory-Eintrag sparen wollen, XCONTROL (und NVDICONF.CPX) nicht benutzen und nur beim Systemstart eine bestimmte Konfiguration sicherstellen möchten, können Sie mit NVDIDFLT dauerhaft die NVDI-Standardeinstellungen setzen. Beim Aufruf des Programms erscheint die folgende Dialogbox:



Nachdem Sie „Auslesen“ aktiviert und im Fileselektor das NVDI.PRG ausgewählt haben, können Sie in der Dialogbox die Standardeinstellungen für NVDI setzen und sichern:



Neben den aus NVDICONF bekannten Schaltern findet sich im NVDIDFLT ein neuer Schalter: „Speicher aneignen“. Ist dieser Schalter aktiviert, so verbiegt NVDI beim Anfordern von VDI-Speicher den Prozeßdeskriptor. Dadurch gehört dieser Speicher NVDI und nicht dem aktuellen Prozess. Sinn dieser Option ist es, Probleme bei Abstürzen (-> Programmentwicklung) und Probleme bei GDOS-Fonts benutzenden Accessories im Zusammenhang mit dem AC_CLOSE-Fehler des TOS (tritt mit Ausnahme von KAOS in TOS-Versionen < 2.05 auf) zu verhindern oder zumindest abzumildern.

Achtung: Da das Verbiegen des Prozeßdeskriptors von ATARI nicht dokumentiert ist, kann es möglicherweise irgendwann in einer neueren TOS-Version zu Problemen beim Verbiegen des Prozeßdeskriptors durch NVDI kommen. Wir empfehlen daher diese Option nicht zu aktivieren (das ist auch die Voreinstellung), es sei denn, Sie befinden sich in der Programmentwicklungsphase oder benutzen eine alte TOS-Version.

GEM_TEST

Mit dem Programm GEM_TEST haben Sie ein Programm zur Hand, das es Ihnen gestattet, verschiedene VDI-Bildschirmtreiber und die Objektausgaberoutine des AES auf ihre Effizienz zu testen. GEM_TEST überprüft dazu außer den Eingabefunktionen fast alle VDI-Funktionen.
Dabei wird berücksichtigt, daß in der Praxis bei den Ausgabefunktionen hauptsächlich drei Zustände auftreten:

1. Zeichnen oder Verschieben kleiner Objekte, z.B. Buttons.
2. Zeichnen oder Verschieben von Teilbereichen, z.B. beim Redraw nach dem Schließen einer Dialogbox.
3. Zeichnen oder Verschieben großer Objekte, z.B. des Desktop-Hintergrundes.

Die Messung kann durch Anklicken der einzelnen Buttons oder durch den Button „Alles testen" gestartet werden. Die Ergebnisse werden relativ zum gewählten Referenzsystem (entspricht 100%) errechnet. Besitzer eines ATARI mit Blitter können über den „Blitter"-Schalter den Blitter ein-/ausschalten und dabei beide Meßreihen betrachten.

Der Dialog „Systeminfo" enhält nähere Informationen über den vorhandenen (Co-)Prozessor, das verwendete GDOS und die Fähigkeiten der Grafik-Hardware.

Die Meßergebnisse können durch Anklicken des Buttons „Meßergebnisse sichern" in einer Datei gesichert werden. In dieser Datei finden Sie neben den prozentualen Ergebnissen auch Einzelmeßwerte in der bei Grafikprozessoren üblichen Einheit Pixel pro Sekunde [P/s]. Außerdem sind in dieser Datei die aus dem Dialog „Systeminfo" bekannten Informationen vermerkt.

Was ist GDOS?

GDOS („*Graphic Device Operating System*“) ist eine standardisierte Schnittstelle des Grafiksystems.

GDOS hat mehrere Aufgaben zu erfüllen:

1. Verwaltung der Gerätetreiber

GDOS lädt und verwaltet die in der ASSIGN.SYS-Datei angemeldeten und von Programmen angeforderten Treiber für grafische Ein- und Ausgaben auf verschieden Geräten.
Diese Gerätetreiber sind Programme, welche die vom GDOS übermittelten VDI-Aufrufe auf einem bestimmten Ausgabegerät (Bildschirm, Plotter, Drucker,...) abbilden.

2. Umrechnen der Koordinatensysteme

Das VDI kennt zwei Koordinatensysteme:

RC-System: Der Nullpunkt liegt in der linken oberen Ecke. Die Rasterpunkte können exakt innerhalb der Geräteauflösung angesprochen werden. Das RC-System wird auch vom AES benutzt.

NDC-System: Der Nullpunkt befindet sich in der linken unteren Ecke. Dem NDC-System liegt immer eine Auflösung von 32768*32768 Punkten zugrunde, wobei diese normalisierten Gerätekoordinaten vom GDOS auf die Auflösung des jeweiligen Ausgabegerätes umgerechnet werden.
Die Nachteile dieser Umrechnung sind Verlangsamung der Ausgabe und mögliche Verzerrungen aufgrund verschiedener Seitenverhältnisse von Monitor und Drucker. Außerdem beherrscht nicht jeder Treiber dieses Koordinatensystem.

3. Verteilung der VDI-Aufrufe an die betreffenden Gerätetreiber

GDOS verteilt verschiedene VDI-Aufrufe an die gewünschten Gerätetreiber, um z.B. Ausgaben sowohl auf Bildschirmen als auch auf Druckern oder sonstigen per Treiber ansprechbaren Geräten durchführen zu können.

4. Verwaltung der Zeichensätze

Die in der ASSIGN.SYS-Datei angemeldeten und einem Gerätetreiber zugeordneten Zeichensätze werden bei Bedarf geladen und dem betreffenden Treiber zur Verfügung gestellt.

Die Benutzung des VDI hat i.a. den Vorteil, daß Ausgabedaten geräteunabhängig erstellt werden können und auf jedem Gerät mit höchstmöglicher Qualität ausgegeben werden können.

Die ASSIGN.SYS-Datei

Die Konfiguration des VDI/GDOS-Systems kann in weiten Grenzen mit der ASSIGN.SYS-Datei beeinflußt werden. Bei der ASSIGN.SYS-Datei handelt es sich um eine ASCII-Datei die mit einem beliebigen Texteditor bearbeitet werden kann. Eine typische ASSIGN.SYS-Datei hat folgenden Aufbau:

```
;
; Beispiel-ASSIGN.SYS für NVDI
; Diese Datei muß im Wurzelverzeichnis des Startlaufwerks liegen!
;
; Das vorangestellte 's' dient zum Einbinden neuer Systemfonts. Auf der NVDI-
;Disk befinden sich die Fonts MONACO und BLUE10. Alle weiteren Fonts stammen
;aus dem Lieferumfang anderer GEM- Applikationen und sind nicht auf der NVDI-
;Disk enthalten.
;
PATH = C:\GEMSYS\                ;Pfadangabe

01p SCREEN.SYS                  ;aktuelle Auflösung
02p SCREEN.SYS                  ;niedrige ST-Auflösung (320*200)
s MONACO08.FNT                  ;Monaco    8 Punkte (Systemfont)
s MONACO09.FNT                  ;Monaco    9 Punkte (Systemfont)
s MONACO10.FNT                  ;Monaco   10 Punkte (Systemfont)
s MONACO20.FNT                  ;Monaco   20 Punkte (Systemfont)

03p SCREEN.SYS                  ;mittlere ST-Auflösung (640*200)
s MONACO08.FNT                  ;Monaco    8 Punkte (Systemfont)
s MONACO09.FNT                  ;Monaco    9 Punkte (Systemfont)
s MONACO10.FNT                  ;Monaco   10 Punkte (Systemfont)
s MONACO20.FNT                  ;Monaco   20 Punkte (Systemfont)
ATSS10CG.FNT
ATSS12CG.FNT
ATSS14CG.FNT
ATSS18CG.FNT
ATSS24CG.FNT

ATTR10CG.FNT
ATTR12CG.FNT
ATTR14CG.FNT
ATTR18CG.FNT
ATTR24CG.FNT

04p SCREEN.SYS                  ;hohe ST-Auflösung (640*400)
s MONACO08.FNT                  ;Monaco    8 Punkte (Systemfont)
s MONACO09.FNT                  ;Monaco    9 Punkte (Systemfont)
s MONACO10.FNT                  ;Monaco   10 Punkte (Systemfont)
```

```
s MONACO20.FNT                       ;Monaco   20 Punkte (Systemfont)

ATSS10.FNT
ATSS12.FNT
ATSS14.FNT
ATSS18.FNT
ATSS24.FNT

ATTR10.FNT
ATTR12.FNT
ATTR14.FNT
ATTR18.FNT
ATTR24.FNT

06p SCREEN.SYS                      ;mittlere TT-Auflösung (640*480)
s MONACO08.FNT                       ;Monaco    8 Punkte (Systemfont)
s MONACO09.FNT                       ;Monaco    9 Punkte (Systemfont)
s MONACO10.FNT                       ;Monaco   10 Punkte (Systemfont)
s MONACO20.FNT                       ;Monaco   20 Punkte (Systemfont)

ATSS10.FNT
ATSS12.FNT
ATSS14.FNT
ATSS18.FNT
ATSS24.FNT

ATTR10.FNT
ATTR12.FNT
ATTR14.FNT
ATTR18.FNT
ATTR24.FNT

08p SCREEN.SYS                       ;hohe TT-Auflösung (1280*960)
s MONACO08.FNT                       ;Monaco    8 Punkte (Systemfont)
s MONACO09.FNT                       ;Monaco    9 Punkte (Systemfont)
s MONACO10.FNT                       ;Monaco   10 Punkte (Systemfont)
s MONACO20.FNT                       ;Monaco   20 Punkte (Systemfont)

09p SCREEN.SYS                      ;niedrige TT-Auflösung (320*480)
s MONACO08.FNT                       ;Monaco    8 Punkte (Systemfont)
s MONACO09.FNT                       ;Monaco    9 Punkte (Systemfont)
s MONACO10.FNT                       ;Monaco   10 Punkte (Systemfont)
s MONACO20.FNT                       ;Monaco   20 Punkte (Systemfont)

21 NECP6.SYS                         ;Druckertreiber für NEC P6/P7
```

```
ATSS10NC.FNT
ATSS12NC.FNT
ATSS14NC.FNT
ATSS18NC.FNT
ATSS24NC.FNT

ATTR10NC.FNT
ATTR12NC.FNT
ATTR14NC.FNT
ATTR18NC.FNT
ATTR24NC.FNT

31 META.SYS                ;Metafiletreiber
ATSS10MF.FNT               ;Metafilefonts
ATSS12MF.FNT
ATSS18MF.FNT
ATSS24MF.FNT
ATTP10MF.FNT
ATTR10MF.FNT
ATTR12MF.FNT
ATTR18MF.FNT
ATTR24MF.FNT
```

Alles, was innerhalb einer Zeile rechts von einem Semikolon steht, wird als Kommentar aufgefaßt, der zum besseren Verständnis der verschiedenen Eintragungen in der ASSIGN.SYS-Datei dienen soll.

Hinter dem (optionalen) Bezeichner '**PATH =** ' wird der Ordner angegeben, der alle Zeichensätze und Gerätetreiber enthält, die Sie in GEM-Applikationen benötigen. Obwohl die Angabe dieses Ordners nicht zwingend ist, ist sie doch ratsam, da sich sonst alle Gerätetreiber und Zeichensätze im Wurzelverzeichnis des Startlaufwerks befinden müssen, was sehr schnell zu großer Unübersichtlichkeit führen kann.
Die Angabe des Ordners über '**PATH =** ' darf nur einmal innerhalb der ASSIGN.SYS-Datei erfolgen.

In den nachfolgenden Zeilen werden die Gerätetreiber und die ihnen zugeordneten Zeichensätze aufgeführt.

Hinweis: Es gibt noch einige ältere GEM-Applikationen, die nur mit einer beschränkten Anzahl von Zeichensätzen arbeiten können. Dies ist keine Beschänkung, die durch VDI/GDOS vorgegeben ist. In so einem Fall sollten Sie für diese Programme eine spezielle ASSIGN.SYS-Datei anlegen und diese bei Bedarf beim Systemstart mit einem Bootutility (z.B. XBOOT) aktivieren. Neuere GEM-Applikationen weisen diese unnötige Beschränkung i.a. nicht auf.

Der Dateiname eines Gerätetreibers muß die Endung '**.SYS**' haben - kommen Sie jetzt jedoch bitte nicht auf die Idee, Ihren möglicherweise ebenfalls mit dieser Endung versehenen Festplattentreiber in der ASSIGN.SYS-Datei einzutragen! Grafikausgaben werden Sie ihm nicht entlocken können.

Vor dem Dateinamen eines Gerätetreibers befindet sich eine zweistellige Zahl, die sogenannte Gerätekennung. Diese Zahl ist grundsätzlich eine Zuordnung zu einem Ausgabegerät, wobei die folgenden Belegungen vordefiniert sind:

01-10	Bildschirmtreiber
11-20	Plottertreiber
21-30	Druckertreiber
31-40	Metafiletreiber
41-50	Kameratreiber
51-60	Grafiktablett-Treiber
ab 61	Memory-Treiber

Achtung: Dadurch, daß Sie z.B. einen Metafiletreiber unter der Nummer eines Druckertreibers eintragen, wird dieser NICHT zu einem Druckertreiber, d.h. man muß einen Gerätetreiber immer einer zugehörigen Gerätekennung zuordnen!
Des weiteren darf eine Gerätekennung in der ASSIGN.SYS-Datei nicht mehrfach benutzt werden (das kann zu unerwarteten Ergebnissen führen).
Es ist anzumerken, daß z.B. viele Programme einen Gerätetreiber nur dann akzeptieren und benutzen, wenn er unter der für den Gerätetyp erstmöglichen Gerätekennung eingetragen ist, d.h. Gerätekennung 21 für einen Druckertreiber und 31 für einen Metafiletreiber (usw.). Die für Programmierer korrekte Vorgehensweise ist, zu versuchen alle einem Gerätetyp zugehörigen Gerätekennungen anzusprechen, bis ein passender Gerätetreiber gefunden wird. Falls kein passender Gerätetreiber vorhanden ist, sollte eine Fehlermeldung ausgegeben werden.

Wie Sie der abgebildeten ASSIGN.SYS-Datei entnehmen können, ist ein Großteil der Bildschirmtreiber kennungen beim ATARI TT bereits mit verschiedenen Auflösungen belegt.

Direkt hinter der Gerätetreiberkennung kann sich einer der folgenden Buchstaben befinden:

'P': Abkürzung für „permanent". Falls NVDI aktiv ist, wird automatisch der passende NVDI-Bildschirmtreiber geladen. Andernfalls wird das ATARI-VDI, welches im ROM vorliegt, benutzt.

'R': Abkürzung für „resident". Der Gerätetreiber wird beim Systemstart geladen und verbleibt resident im Speicher.

Befindet sich kein Buchstabe hinter der Gerätekennung, so wird der Gerätetreiber beim Öffnen geladen und beim Schließen wieder aus dem Speicher entfernt.

Hinweis: Die NVDI-Bildschirmtreiber sollten nicht mit Ihrem Namen in der ASSIGN.SYS-Datei eingetragen werden, da es sonst bei der Zusammenarbeit mit anderen GDOSsen zu Problemen kommen kann. Solange kein GDOS-Treiber für eine Grafikkarte benutzt wird, genügt der Eintrag 'p SCREEN.SYS' hinter der Gerätekennung.

Unterhalb des Gerätetreibers werden die ihm zugeordneten Fonts (Endung des Dateinamens muß '.FNT' sein) eingetragen. Das bedeutet, daß beispielsweise bei einem Bildschirmtreiber nur die ihm in der jeweiligen Auflösung, also nur die unter der Gerätekennung eingetragen, Fonts zur Verfügung stehen und Speicher belegen.
Ähnlich wie die Gerätetreiber werden normalerweise auch die Fonts bei Bedarf geladen und dann wieder aus dem Speicher entfernt. Zwei Buchstabenkennungen bieten jedoch noch Modifikationsmöglichkeiten:

'R ': Ein dem Fontnamen vorangestelltes 'R ' (mit Leerzeichen!) sorgt dafür, daß der Font beim Systemstart geladen wird und resident im Speicher verbleibt.

'S ': Beim NVDI (und nur hier) bewirkt ein dem Dateinamen vorangestelltes **'S '** (mit Leerzeichen!) eine Einbindung als Systemfont. So können Sie bei Bedarf die Systemfonts des ATARI ersetzen (die bedeutet alle, nicht nur den 10-Punkt-Font!). Wichtig dabei ist nur, daß ein so angemeldeter Font von den Außmaßen und der Anzahl der Zeichen (256) dem zu ersetzenden Systemfont entspricht.
Das Ersetzen eines oder mehrerer Systemfonts bietet sich an, wenn Sie z.B. einen vollständig IBM-kompatiblen Font benötigen und erspart Ihnen einen Hardware-Eingriff ins Betriebssystem.

NVDI bietet ab der Version 2.00 die Möglichkeit, mit der Kennung **'S '** auch auf dem ATARI ST einen vierten Systemfont (Höhe 20pt) einzubinden. Dies kann mit fehlerhafter Software (z.B. Wordplus-Versionen vor 3.15a) zu Problemen führen, die durch Einschalten der Fehlerkompatibilität oder Entfernen des 20pt-Fonts aus der ASSIGN.SYS-Datei behoben werden können.

Hinweis: Das residente Laden eines Treibers oder Zeichensatzes hat den Nachteil, daß der dabei belegte Speicher erst beim Neustart des Rechners wieder freigegeben wird und ist daher in der Regel nur dann sinnvoll, wenn ausschließlich mit einem Diskettensystem gearbeitet wird. Die Geschwindigkeit der heutigen Festplattensysteme und der Speicherverbrauch sprechen meist gegen den Einsatz dieser Option.

Eintragungen in der ASSIGN.SYS-Datei können übrigens in Groß- oder Kleinschrift vorgenommen werden. Solange die Angaben syntaktisch korrekt sind, beinträchtigt dies die Funktion von NVDI nicht. Auf die Verwendung von Umlauten bei Dateinamen sollten Sie jedoch verzichten, da das Betriebssystem damit Probleme haben könnte.

Achtung: Wenn Sie NVDI ohne ASSIGN.SYS-Datei betreiben, dann müssen sich die NVDI-Treiber (NVDIDRVx.SYS) im Hauptverzeichnis des Startlaufwerks befinden. Andernfalls in dem unter **'PATH = '** angegebenen Verzeichnis.

Zeichensätze

Beim Anblick der abgedruckten ASSIGN.SYS-Datei wird bei Ihnen sicher die Frage auftauchen, was sich hinter einer Bezeichnung wie ATSS10NC.FNT verbirgt.
Die Idee hinter dieser Bezeichnung wird schnell klar, wenn man den Namen in vier Bereiche aufteilt:

ccttppdd.FNT

<cc> - Kürzel für den Hersteller des Zeichensatzes. Bisher bekannte Kürzel sind:

AT	- Atari
2B	- unser Kürzel

<tt> - Kürzel für das Schriftbild. Bisher bekannte Kürzel sind:

SS	- Sans Serif; ATARIs „Swiss"-Zeichensätze
TP	- Typewriter;
TR	- Times Roman; ATARIs „Dutch"-Zeichensätze

<pp> - Die Größe des Zeichensatzes in Punkten (1/72")

<dd> - Kürzel für das Gerät, für das der Zeichensatz erstellt wurde. Bisher bekannte Kürzel sind:

CG	- Zeichensatz für die mittlere ST-Auflösung (aufgrund der nicht quadratischen Pixelgröße sind hier spezielle Zeichensätze erforderlich.
EP	- Epson 9-Nadeldrucker (120 * 144 dpi)
SP	- Star NB 15 24-Nadeldrucker (180 * 180 dpi)
LQ	- Epson LQ-Serie 24-Nadeldrucker (180 * 180 dpi)
LS	- Laserdrucker (300 * 300 dpi)
NC	- NEC P-Serie 24-Nadeldrucker (360 * 360 dpi)
MF	- Metafile

Ist das Gerätekürzel unbenutzt, dann handelt es sich um Zeichensätze für Bildschirmauflösungen mit quadratischen Pixelgrößen (ST Hoch, ST Niedrig,...).

Wenn Sie Zeichensatznamen nach diesem Schema ändern, müssen Sie beachten, daß GDOS die Zeichensätze nicht anhand des Namens, sondern einer internen Kennung unterscheidet. Es dürfen also auf keinen Fall Zeichensatzfamilien mit unterschiedlichem Namen aber gleicher interner Kennung auftreten. Das könnte entweder GDOS oder die jeweilge Applikation ziemlich verwirren (einer der Gründe, weshalb wir bei den inzwischen recht weit verbreiteten MONACO- und BLUE-Zeichensätzen auf eine Namensänderung verzichtet haben).

Fragen

Sie finden hier in loser Reihenfolge die beliebtesten Fragen an den Software-Support.

?: „Mir gefallen die mitgelieferten Systemfonts nicht. Wie bekomme ich wieder die Orginalfonts auf den Schirm?“

A: Indem Sie mit einem Texteditor die ‘S ‘-Kennung vor den Fonts in der ASSIGN.SYS-Datei entfernen.

?: „Wie muß ein Font aufgebaut sein, damit ich ihn als Systemfont einbinden kann?“

A: Der GEM-Font muß aus 256 äquidistanten Zeichen bestehen und in seinen Ausmaßen dem zu ersetzen ATARI-Systemfont entsprechen.

?: „Was kann ich mit der ASSIGN.SYS-Datei anfangen?“

A: In der ASSIGN.SYS-Datei sind die Gerätetreiber und Fonts eingetragen, die NVDI verwaltet. Näheres erfahren Sie im Kapitel „Die ASSIGN.SYS-Datei“.

?: „Welchen Sinn haben die Dateien im GEMSYS-Ordner?“

A: Üblicherweise kopiert man Gerätertreiber und Fonts in den GEMSYS-Ordner. Sinn des Ganzen ist es, etwas Ordnung zu schaffen. Weitere Informationen entnehmen Sie bitte dem Kapitel „Die ASSIGN.SYS-Datei“.

?: „Welche OverScan-Version benötige ich zusammen mit NVDI?“

A: Um problemlos mit NVDI und OverScan zu arbeiten, sollten Sie mindestens die Version 3.0zd verwenden.

?: „Beschleunigt NVDI auch die Druckerausgabe?“

A: NVDI beschleunigt die Bildschirmausgabe und nicht die Druckerausgabe. Nur ein schnellerer Druckertreiber kann die Druckerausgabe beschleunigen.

?: „Wieso kann ich keine SIGNUM!-Fonts in der ASSIGN.SYS-Datei einbinden?“

A: Die SIGNUM!-Fonts liegen in einem nicht GEM-konformen Format vor. Sie können erst nach einer Konvertierung als GEM-Fonts benutzt werden.

?: „Wieso beschleunigt NVDI nicht SIGNUM!?“

A: SIGNUM! benutzt eigene Ausgaberoutinen. Wegen dieser Umgehung des Betriebssystems durch SIGNUM! kann NVDI die Ausgaben nicht beschleunigen.

?: „Wie komme ich an Druckertreiber und Zeichensätze?“

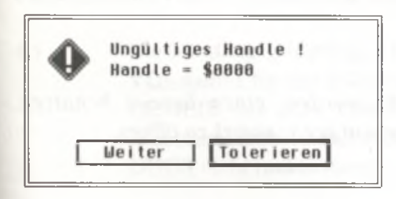
A: Üblicherweise liegen sie GDOS-unterstützenden GEM-Applikationen (z.B. SciGraph) bei. Zum Teil sind Gerätetreiber und Fonts auch in Mailboxen (z.B. in der MAUS) per Modem abrufbar.

?: „Wieso verträgt sich NVDI nicht zusammen mit anderen VDI-Beschleunigern?“

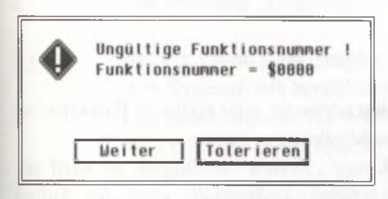
A: Mal abgesehen von dem Sinn so einer Aktion müßte die Frage eigentlich lauten: „Wieso vertragen sich andere VDI-Beschleuniger nicht mit NVDI?“. Die Antwort: Weil sie direkt auf undokumentierte Variablen des ATARI-VDI zugreifen, die beim NVDI und beim ATARI TT ihre Lage verändert haben.

Fehlermeldungen

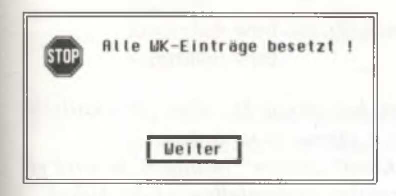
Bei fehlerhaften VDI-Aufrufen, falschen Angaben in der ASSIGN.SYS-Datei oder Speicherplatzmangel kann eine Alert-Box mit einer der folgenden Meldungen erscheinen.



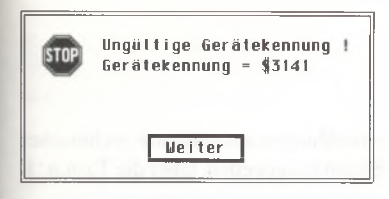
Das VDI wurde mit einem fehlerhaften Handle aufgerufen. Durch Betätigen des „Tolerieren"-Knopfes können Sie die Fehlerkompatibilität einschalten. Andernfalls wird der VDI-Aufruf ignoriert.



Einer der NVDI-Gerätetreiber wurde mit einer nicht vorhandenen Funktionsnummer aufgerufen.
Durch Betätigung des Knopfes „Tolerieren" können Sie die Fehlerkompatibilität einschalten.



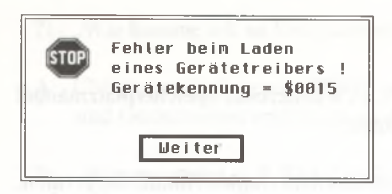
Es sind bereits 128 Workstations geöffnet worden. Möglicherweise hat ein mehrmals aufgerufenes Programm v_clsvwk nicht benutzt.



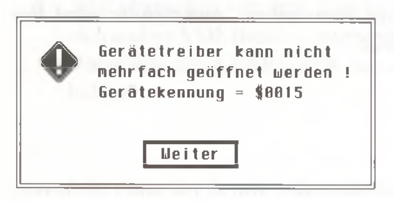
Eine ungültige Gerätekennung (größer 99) wurde bei v_opnwk in work_in[0] übergeben.



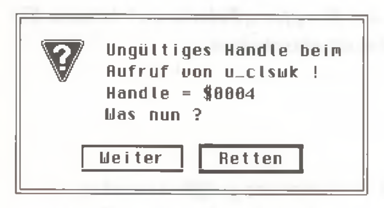
Es ist nicht genügend Speicher vorhanden, um eine Workstation zu öffnen.



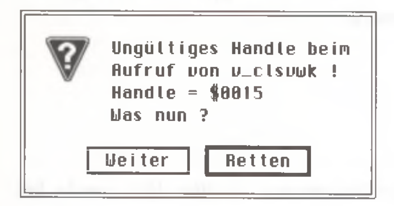
Diese Meldung wird ausgegeben, wenn beim v_opnwk das Laden eines Gerätetreibers nicht möglich ist, was aufgrund von Speichermangel oder eines fehlerhaften Dateinamens geschehen kann.



Es ist versucht worden, einen bereits benutzen Gerätetreiber erneut per v_opnwk zu öffnen.



Ein Programm hat versucht, eine virtuelle Workstation mit v_clswk zu schließen.
Wenn Sie den Knopf „Retten“ betätigen, so wird ein v_clsvwk durchgeführt. Andernfalls wird der Aufruf ignoriert.



Ein Programm hat versucht, eine physikalische Workstation mit v_clsvwk zu schließen.
Wenn Sie den Knopf „Retten“ betätigen, so wird ein v_clswk durchgeführt. Andernfalls wird der Aufruf ignoriert.

Wenn NVDI mit FSMGDOS zusammenarbeitet, werden die Fehlermeldungen aus software-technischen Gründen in der obersten Bildschirmzeile und nicht über ALERT-Boxen ausgegeben. Über die Tasten '1' und '2' können Sie dann das Betätigen der Knöpfe simulieren.

Fehlerkompatibilität

NVDI enthält für Programme, die unsauber programmiert wurden, einen Schalter, der verschiedene NVDI-Funktionen so einschränkt, daß sie im Funktionsumfang dem ATARI-VDI entsprechen. Bei eingeschalteter Fehlerkompatibilität ändert sich das Verhalten von NVDI wie folgt:

1.) Die Meldung „Ungültiges Handle!" wird unterdrückt. Dabei wird vorausgesetzt, daß der VDI-Aufruf für den Bildschirmtreiber bestimmt war. Man findet diesen Fehler z.B. in der GEMLIB des alten OMIKRON-Basic. Software mit diesem Fehler wird mit großer Wahrscheinlichkeit mit anderen Bildschirmtreibern (Großbildschirm) oder einem anderen GDOS nicht funktionieren.

2.) Die Meldung „Ungültige Funktionsnummer" wird nicht ausgegeben.

3.) Bei vst_point(), vst_height(), v_opnwk(), v_opnvwk() und vq_extnd() werden nur die ersten drei Systemfonts berücksichtigt.
Das geschieht z.B. für WORDPLUS-Versionen < 3.15a auf dem TT. Dieses Verhalten von NVDI ist nur eine „Krücke" für fehlerhafte Programme.

4.) Die Funktion vst_height() wird für den Systemzeichensatz auf eine Höhe von 26 Pixeln begrenzt.
Zusätzlich wird das „Skalierungsverhalten" geändert, so daß - wenn möglich - immer nur vergrößert wird.

5.) Die Funktion „vqt_attributes" liefert in attrib[5] den Schreibmodus -1 zurück. Dies entspricht zwar nicht der VDI-Dokumentation aber dem Verhalten des ATARI-ROM-VDI.

Fehlersuche

Saubere GEM-Applikationen sollten mit der Standard-Konfiguration von NVDI problemlos laufen:

- GDOS, GEMDOS-Zeichenausgabe, Maus-Beschleuniger und Line-A an
- Fehlerkompatiblität aus

Für Probleme, die zusammen mit NVDI und einem bestimmten Programm auftreten, kann es folgende Erklärungen geben:

1.) Eines Ihrer Accessories oder AUTO-Ordner-Programme versucht, auf undokumentierte interne VDI-Variablen zuzugreifen, unzulässige Modifikationen in den Line-A-Variablen durchführen oder Systemvektoren zu „verbiegen".
Durch schrittweises Aktivieren und Deaktivieren von Accessories und AUTO-Ordner-Programmen können Sie die problemträchtige Kombination ermitteln.

2.) Die betreffende Applikation arbeitet nicht mit GDOS zusammen. Wenn dies der Fall ist, müssen Sie das GDOS im NVDICONF deaktivieren.
Sollte die Software nach dieser Maßnahme einwandfrei funktionieren, so wenden Sie sich bitte mit einer Fehlerbeschreibung an den Hersteller.

3.) Die Applikation kommt mit den erweiterten Fähigkeiten von NVDI (leistungsfähigere Textvergrößerung, 4. Systemfont...) nicht zurecht.
Wenn sich die Applikation nach Einschalten der Fehlerkompatibilität anstandslos benutzen läßt, sollten Sie sich unter Angabe des Fehlers mit dem Hersteller der Software in Verbindung setzen.

4.) Die Applikation benutzt Line-A-Funktionen. In diesem Fall sollten Sie im NVDICONF Line-A einschalten.

5.) Das Programm sichert bei GEMDOS-Ausgaben nicht genügend Register. Wenn es nach Ausschalten der schnellen GEMDOS-Ausgabe problemlos läuft, sollten Sie sich unter Angabe des Fehlers an den Hersteller wenden.

Sollte nach diesen Schritten das Problem immer noch vorhanden sein, so wenden Sie sich bitte schriftlich unter Angabe Ihrer registrierten Programmnummer an BELA Computer GmbH.

Das VDI für Programmierer

Die folgenden Abschnitte wenden sich im wesentlichen an den Programmierer. Sie finden neben der Beschreibung der VDI-Funktionen und der NVDI-Schnittstelle nützliche VDI-Programmiertips. Den Abschluß bildet eine Übersicht der in den NVDI-Bildschirmtreibern implementierten Funktionen und einen Index der VDI-Funktionen.

Die innere Struktur von NVDI ist in acht Gruppen aufteilbar:

0. GDOS-Funktionen
1. Kontrollfunktionen
2. Ausgabefunktionen
3. Attributfunktionen
4. Rasteroperationen
5. Eingabefunktionen
6. Auskunftsfunktionen
7. ESCAPE-Funktionen

Nachfolgend werden die Funktionen, ihre Deklaration, die einzelnen Funktionsparameter und der Aufruf in ANSI-C beschrieben.
Weiterhin werden die Beschränkungen und Reaktionen der Routinen bei auftretenden Fehlern aufgeführt und bestehende oder in Zukunft mögliche (Fehler-) Kompatibilitätsprobleme erwähnt.

Aufruf der VDI-Funktionen/Parameterübergabe

Zur Programmierung des VDI werden fünf Arrays benutzt:

contrl[]	:	Opcode-spezifische Informationen (Anzahl der Parameter, ...)
intin[]	:	Eingabe von Integers
ptsin[]	:	Eingabe von Koordinaten
intout[]	:	Ausgabe von Integers
ptsout[]	:	Ausgabe von Koordinaten

Das VDI benutzt drei Arten von Parametern: Eingabe-, Ausgabe-und Ein-/Ausgabe-Parameter. Diese sind den Arrays wie folgt zugeordnet:

1. Eingabeparameter (müssen vom Anwenderprogramm gesetzt werden)

contrl[0]	:	Funktionsnummer
contrl[1]	:	Anzahl der Koordinatenpaare in ptsin
contrl[3]	:	Anzahl der Integers in intin
contrl[5]	:	Unterfunktionsnummer
intin[]	:	Eingabe-Array für Integers
ptsin[]	:	Eingabe-Array für Koordinaten

2. Ausgabeparameter (werden vom Gerätetreiber bestückt)

contrl[2] : Anzahl der Koordinatenpaare in ptsout
contrl[4] : Anzahl der Integers in intout
intout[] : Ausgabe-Array für Integers
ptsout[] : Ausgabe-Array für Koordinatenpaare

3. Ein-/Ausgabeparameter:

contrl[6] : Handle (wird von v_opnwk/v_opnvwk eingetragen und danach als Eingabeparameter bei Funktionsaufrufen benutzt)
contrl[7..11]: Belegung und Benutzung funktionsabhängig.

Man beachte, daß in contrl[1] die Anzahl der Paare in ptsin angegeben wird, d.h. die Anzahl der Feldelemente dividiert durch zwei.

In der Anleitung entspricht die Deklaration der VDI-Funktionen dem bei C-Compilern üblichen Standard. Deren Bibliotheken kopieren z.B. eigenständig die Rückgabewerte aus den verschiedenen VDI-Arrays in das dem Compiler übergebene Array. Bei anderen Sprachen (Basic, Assembler, etc.) müssen Sie i.a. die Rückgabewerte selbst aus den verschiedenen Arrays entnehmen.

Beispiel: Die Funktion „*vst_height*“. Diese gestattet es, die Zeichenhöhe von der Basislinie bis zur Zeichenzellenobergrenze festzulegen.

Die C-Deklaration lautet:

*void vst_height (int handle, int height, int *char_width, int *char_height, int *cell_width, int *cell_height);*

Der Aufruf erfolgt dann so:

vst_height(handle, height, &char_width, &char_height, &cell_width, &cell_height);

Als Rückgabewerte erhält man:

char_width : ausgewählte Zeichenbreite
char_height : ausgewählte Zeichenhöhe
cell_width : ausgewählte Zeichenzellenbreite
cell_height : ausgewählte Zeichenzellenhöhe

Als Beispiel für eine andere höhere Programmiersprache sei (aufgrund der Verbreitung) hier GFA-BASIC 3.x gewählt.
Zuerst wollen wir die für diese Funktion notwendigen Parameter betrachten:

Variable	Belegung	Bedeutung
contrl[0]	12	vst_height
contrl[1]	1	Einträge in ptsin
contrl[2]	2	Einträge in ptsout
contrl[3]	0	Einträge in intin

contrl[4]	0	Einträge in intout
contrl[6]	handle	
ptsin[1]	height	gewünschte Zeichenhöhe
ptsout[0]	char_width	ausgewählte Zeichenbreite
ptsout[1]	char_height	ausgewählte Zeichenhöhe
ptsout[2]	cell_width	ausgewählte Zeichenzellenbreite
ptsout[3]	cell_height	ausgewählte Zeichenzellenbreite

In GFA-BASIC müßte man folgendermaßen vorgehen:

1. Eine PROCEDURE erstellen, welche die notwendigen Parameter setzt, das VDI aufruft und anschließend die Rückgabewerte umkopiert.

```
PROCEDURE vst_height(height&,VAR char_width&,char_height&,cell_width&, cell_height&)
    PTSIN(0)=0
    PTSIN(1)=height&              !gewünschte Zeichenhöhe
    CONTRL(0)=12                  !VDI-Opcode für „vst_height“
    CONTRL(1)=1                   !Anzahl der Einträge in PTSIN
    CONTRL(3)=0                   !Anzahl der Einträge in INTIN
    VDISYS                        !Das Handle (contrl[6]) wird von GFABASIC gesetzt
    char_width&=PTSOUT(0)         !Jetzt folgt das Kopieren der Rückgabewerte
    char_height&=PTSOUT(1)
    cell_width&=PTSOUT(2)
    cell_height&=PTSOUT(3)
RETURN
```

2. Nun die VDI-Funktion durch

vst_height(height&,char_width&,char_height&,cell_width&,cell_height&)

aufrufen. Die Rückgabewerte entsprechen denen der C-Routine.

Die GDOS-Funktionen

VQ_GDOS

Deklaration: int vq_gdos(void);
Aufruf: exist = vq_gdos();

Bedeutung von exist:
0: kein GDOS
<> 0: GDOS vorhanden

Die Funktion ermittelt, ob GDOS installiert ist. Falls GDOS nicht installiert ist, liefert sie Null zurück, andernfalls einen von Null verschiedenen Rückgabewert.

Implementierung in Assembler:

```
vq_gdos: moveq.l #-2,d1
trap #2
subq.w #2,d0
sne.b d0
ext.w d0
rts
```

V_SET_APP_BUF

Mit dieser Funktion kann der für Bezier-Generierung zur Verfügung stehende Buffer vergrößert werden. NVDI verwendet intern einen Buffer von 16kB und übernimmt den bei v_set_app_buff() angegebenen Buffer nur, wenn er größer als der interne Buffer ist.

Deklaration: void v_set_app_buff(int handle, int *address, int nparagraphs);
Aufruf: v_set_app_buff(handle, address, nparagraphs);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	-1	v_set_app_buff
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	3	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	6	
contrl[6]	handle	
intin[0..1]	address	Bufferadresse
intin[2]	nparagraphs	Bufferlänge/16 = Anzahl der „paragraphs"

Bemerkungen:

Der Buffer wird aus „historischen“ Gründen in Abschnitte („paragraphs“) von jeweils 16 Bytes eingeteilt, was bei der Angabe der Bufferlänge berücksichtigt werden muß. Da die Bezier-Generierung je nach GDOS unterschiedlich sein kann, sind Annahmen über die Struktur der im Buffer abgelegten Daten unzulässig!

In GEM/3 ist diese Funktion so implementiert worden, daß der Buffer global für alle Applikationen verwendet wird und bei dieser Funktion kein Handle übergeben wird, was natürlich ein schwerer Design-Fehler ist und sich spätestens in einem Multitasking-Betriebssystem rächt. Um diesen Design-Fehler zu beseitigen, benutzt NVDI den übergebenen Buffer nur für die mit <handle> angegebene (virtuelle) Workstation.

Nach GEM/3-Dokumentation sollte das Programm, welches den Buffer zur Verfügung gestellt hat, vor dem Beenden die Funktion v_set_app_buff() mit einem Nullpointer als Bufferadresse aufrufen, um die weitere Benutzung dieses Buffers durch GDOS zu verhindern. Da die Buffer in NVDI lokal sind, ist dieser Aufruf nicht notwendig - er schadet aber auch nicht. :-)

Wie Sie sehen, hat diese Funktion einige Haken und Ösen. Um Schwierigkeiten mit anderen GDOSsen (mit Bezier-Unterstützung) im Multitasking-Betrieb zu vermeiden, empfehlen wir Ihnen möglichst auf den Einsatz dieser Funktion zu verzichten und es bei den Standardvorgaben des jeweiligen GDOS zu belassen.

Die VDI-Funktionen

1. Kontrollfunktionen

In diesem Abschnitt finden Sie Funktionen, mit denen Sie die Initialisierung und Grundeinstellung einer VDI-Workstation vornehmen können.
Sie haben hier die „Verwaltung“ des VDI vor sich, deren Aufgabe z.B. darin besteht, die VDI-Befehle auf die unterschiedlichen Gerätetreiber zu verteilen.

OPEN WORKSTATION (VDI 1)

Mit dieser Funktion öffnen Sie eine physikalische Workstation. Dazu wird ein in der ASSIGN.SYS-Datei eingetragener Gerätetreiber geladen und (sofern möglich) entsprechend den Eingabeparametern initialisiert.

Wenn die Initialisierung erfolgreich verlaufen ist, wird in contrl[6] eine Kennung (im weiteren Verlauf Handle genannt) zurückgegeben, andernfalls eine Null.

Falls keine ASSIGN.SYS-Datei vorhanden ist, haben Sie nur Zugriff auf den Bildschirm-Gerätetreiber.

Wichtig: Der Bildschirmtreiber wird nach Abarbeitung des AUTO-Ordners vom AES geöffnet. Anwenderprogramme müssen daher (nach dem Start des AES) zur Bildschirmausgabe eine virtuelle Workstation (VDI 100) öffnen.
Das bedeutet jedoch nicht, daß in AUTO-Ordnerprogrammen keine VDI-Funktionen verwendet werden könnten! Näheres dazu siehe Abschnitt ‘Programmiertips’.

Deklaration: void v_opnwk(int *work_in, int *handle, int *work_out);
Aufruf: v_opnwk(work_int, &handle, work_out);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	1	v_opnwk
contrl[1]	0	Einträge in ptsin
contrl[2]	6	Einträge in ptsout
contrl[3]	11	Einträge in intin
contrl[4]	45	Einträge in intout
contrl[6]	handle	
intin[0..10]	work_in[0..10]	
intout[0..44]	work_out[0..44]	
ptsout[0..11]	work_out[45..56]	

Bedeutung von work_in[0..10]:

work_in[0]: Geräteidentifikationsnummer. Mit ihr wählen Sie den zu ladenden Gerätetreiber.

1-10: Bildschirmtreiber
1: aktuelle Auflösung
2: 320*200, 16 Farben
3: 640*200, 4 Farben
4: 640*400, monochrom
6: 640*480, 16 Farben (TT)
8: 1280*960, monochrom (TT)
9: 320*480, 256 Farben (TT)

ab 11: Plottertreiber
ab 21: Druckertreiber
ab 31: Metafiletreiber
ab 41: Kamera
ab 51: Grafiktablett
ab 61: Memory-Treiber

work_in[1]: Linientyp
work_in[2]: Linienfarbe
work_in[3]: Markertyp
work_in[4]: Markerfarbe
work_in[5]: Zeichensatznummer
work_in[6]: Textfarbe
work_in[7]: Fülltyp
work_in[8]: Füllmuster-Index
work_in[9]: Füllmuster-Farbe
work_in[10]: Koordinatenflag 0: NDC, 2: RC

Bedeutung von work_out[0..56]:

work_out[0]: Adressierbare Rasterbreite (Wertebereich 0 - xmax)
work_out[1]: Adressierbare Rasterhöhe (Wertebereich 0 - ymax)
work_out[2]: Gerätekoordinatenflag
0: genaue Skalierung möglich (z.B. Bildschirm)
1: keine genaue Skalierung möglich (Film-Recorder)
work_out[3]: Breite eines Pixels in Mikrometern
work_out[4]: Höhe eines Pixels in Mikrometern
work_out[5]: Anzahl der Zeichenhöhen (0: beliebig veränderbar)
work_out[6]: Anzahl der Linientypen
work_out[7]: Anzahl der Linienbreiten (0: beliebig veränderbar)
work_out[8]: Anzahl der Markertypen
work_out[9]: Anzahl der Markergrößen (0: beliebig veränderbar)
work_out[10]: Anzahl der verfügbaren Zeichensätze
work_out[11]: Anzahl der Muster
work_out[12]: Anzahl der Schraffuren
work_out[13]: Anzahl der Farben

work_out[14]: Anzahl der GDPs
work_out[15] bis work_out[24]:
Liste der GDPs, deren Ende durch -1 gekennzeichnet ist.
work_out[25] bis work_out[34]:
Liste der Attribute der GDPs:
0: Linie
1: Marker
2: Text
3: ausgefüllter Bereich
4: keine Attribute
work_out[35]: Farbdarstellungsflag
work_out[36]: Textrotationsflag
work_out[37]: Flächenfüllung
work_out[38]: CELLARRAY-Flag
work_out[39]: Anzahl der Farbabstufungen (0: mehr als 32767)
work_out[40]: Kontrolle des Mauszeigers
1: Tastatur
2: Tastatur und Maus (oder anderes Gerät)
work_out[41]: Gerät für variierende Eingaben
1: Tastatur
2: anderes Gerät
work_out[42]: Auswahltasten
1: Funktionstasten
2: anderes Tastenfeld
work_out[43]: String-Eingabe
1: Tastatur
work_out[44]: Geräte-Typ
0: nur Ausgabe
1: Eingabe
2: Ein- u. Ausgabe
4: Metafile-Ausgabe
work_out[45]: geringste Zeichenbreite
work_out[46]: geringste Zeichenhöhe
work_out[47]: größte Zeichenbreite
work_out[48]: größte Zeichenhöhe
work_out[49]: geringste Linienbreite
work_out[50]: 0
work_out[51]: größte Linienbreite
work_out[52]: 0
work_out[53]: geringste Markerbreite
work_out[54]: geringste Markerhöhe
work_out[55]: größte Markerbreite
work_out[56]: größte Markerhöhe

Bemerkung:

Im ATARI-VDI wird aufgrund eines Fehlers als Füllmusterindex nicht der work_in[8]-Wert, sondern work_in[8] + 1 eingestellt. Dieser Fehler wird von NVDI nicht unterstützt.

OPEN VIRTUAL SCREEN WORKSTATION (VDI 100)

„**OPEN VIRTUAL SCREEN WORKSTATION**" öffnet eine virtuelle Bildschirm-Workstation auf einer bereits geöffneten physikalischen Workstation. Dadurch können die Zugriffe verschiedener Programme mit ihren unterschiedlichen Einstellungen koordiniert werden.
Für Bildschirmtreiber müssen Sie das Handle der AES-Bildschirm-Workstation nach der Anmeldung Ihres Programmes beim AES mit

aes_handle = graf_handle(&gr_hwchar,&gr_hhchar,&gr_hwbox,&gr_hhbox);
handle = aes_handle;

ermitteln.

Deklaration: void v_opnvwk(int *work_in, int *handle, int *work_out);
Aufruf: v_opnvwk(work_in, &handle, work_out);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	100	v_opnvwk
contrl[1]	0	Einträge in ptsin
contrl[2]	6	Einträge in ptsout
contrl[3]	11	Einträge in intin
contrl[4]	45	Einträge in intout
contrl[6]	handle	hier Ein-/Ausgabeparameter
intin[0..10]	work_in[0..10]	
intout[0..44]	work_out[0..44]	
ptsout[0..11]	work_out[45..56]	

Die Parameter sind mit denen von „OPEN WORKSTATION" identisch.

CLOSE WORKSTATION (VDI 2)

„**CLOSE WORKSTATION**" schließt eine physikalische Workstation. Vorher sollten alle virtuellen Workstations geschlossen werden.
Bei Druckertreibern werden vor dem Schließen ggf. die noch gepufferten Kommandos ausgeführt, bei Metafiletreibern wird der Metafile geschloßen.

Deklaration: void v_clswk(int handle);
Aufruf: v_clswk(handle);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	2	v_clswk
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	

CLOSE VIRTUAL SCREEN WORKSTATION (VDI 101)

Mit dieser Funktion wird eine geöffnete virtuelle Bildschirm-Workstation geschlossen. Das ATARI-VDI reagiert sehr ungehalten, wenn Sie diese Funktion anwenden und vorher (z.B. durch einen Fehler) keine virtuelle Workstation öffnen konnten.

Deklaration: void v_clsvwk(int handle);
Aufruf: v_clsvwk(handle);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	101	v_clsvwk
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	

CLEAR WORKSTATION (VDI 3)

Diese Funktion löscht den Bildschirm. Bei Plottern oder Druckern wird ein Seitenvorschub durchgeführt und der Druckpuffer gelöscht.

Deklaration: void v_clrwk(int handle);
Aufruf: v_clrwk(handle);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	3	v_clrwk
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	

UPDATE WORKSTATION (VDI 4)

Mit „UPDATE WORKSTATION" werden die gepufferten Kommandos auf einem Gerät (z.B. einem Drucker) ausgeführt, wobei nach der Ausgabe kein Seitenvorschub stattfindet.
Bei Bildschirmtreibern ist diese Funktion unnötig, da Grafikkommandos sofort abgearbeitet werden.

Deklaration: void v_updwk(int handle);
Aufruf: v_updwk(handle);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	4	v_updwk
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	

Bemerkung:
Bei Druckertreibern gibt es die Möglichkeit einen eigenen Buffer auszugeben.

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	4	v_updwk
contrl[1]	1	Buffer nicht löschen
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
intin[0..1]	buffer	Anfangsadresse des Buffers

LOAD FONTS (VDI 119)

Diese Funktion lädt die in ASSIGN.SYS eingetragenen Zeichensätze und gibt deren Anzahl zurück.

Deklaration: int vst_load_fonts(int handle, int select);
Aufruf: additional = vst_load_fonts(handle, 0);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	119	vst_load_fonts
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[6]	handle	
intin[0]	select	0 (reserviert)
intout[0]	additional	Anzahl der geladenen Zeichensätze

Bemerkung:

Es gibt auch eine Möglichkeit, bei ausgeschaltetem oder nicht vorhandenem GDOS zusätzliche Zeichensätze einzubinden (bei aktivem GDOS versagt diese Methode!).
Es empfiehlt sich folgendes Vorgehen (bei den kursiv gestellten Wörter handelt es sich um Elemente der Zeichensatzstruktur s. Abschnitt 'Der Zeichensatz'):

- Mittels vq_gdos abtesten, ob GDOS vorhanden ist. Wenn ja, so rufen Sie vst_load_fonts wie oben beschrieben auf. Andernfalls verfahren Sie weiter nach dem folgenden Muster:
- Speicher für den gewünschten Zeichensatz mit der GEMDOS-Funktion Malloc anfordern.
- Den oder die gewünschten Zeichensätze laden.
- Falls ein Zeichensatz im Intel-Format vorliegt (ist in flags vermerkt), so müssen High- und Low-Byte eines Wortes vertauscht werden. Anschließend müssen bei hor_table, off_table und dat_table High- und Low-Word vertauscht werden.
- Anfangsadresse des Zeichensatzes zu hor_table, off_table und dat_table addieren.
- Adresse des nächsten Fontheaders (oder 0, wenn keine weiteren Zeichensätze folgen) in next_font eintragen.
- Adresse des ersten geladenen Fontheaders in contrl[10..11] eintragen.
-Speicher für Texteffekt-Buffer mit Malloc anfordern (für große Fonts mindestens 2kB).
- Adresse des Buffers in contrl[7..8] übergeben.
- Halbe Länge des Buffers (in Bytes) in contrl[9] eintragen.
- vst_load_fonts aufrufen.

Die so installierten Zeichensätze können wie die von GDOS geladenen behandelt werden.

UNLOAD FONTS (VDI 120)

Der durch die Zeichensätze belegte Speicher wird von „**UNLOAD FONTS**" freigegeben.

Deklaration: void vst_unload_fonts(int handle, int select);
Aufruf: vst_unload_fonts(handle, 0);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	120	vst_unload_fonts
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
intin[0]	select	0 (reserviert)

Bemerkung:
Wenn bei nicht vorhandenem bzw. ausgeschaltetem GDOS Zeichensätze eingebunden worden sind, so muß deren Speicher nach Aufruf von vst_unload_fonts mit der GEMDOS-Funktion Mfree wieder freigegeben werden.

SET CLIPPING RECTANGLE (VDI 129)

Mit dieser Funktion kann man den Arbeitsbereich der Grafikoperationen begrenzen oder freigeben. Ist der Arbeitsbereich begrenzt worden, so werden überstehende Teile nicht ausgegeben.

Deklaration: void vs_clip(int handle, int clip_flag, int *pxyarray);
Aufruf: vs_clip(handle, clip_flag, pxyarray);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	129	vs_clip
contrl[1]	2	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
intin[0]	clip_flag	0: Clipping aus, 1: Clipping an
ptsin[0..3]	pxyarray[0..3]	Arbeitsbereich

Bemerkung:

Das Clipping sollte aus Sicherheitsgründen generell eingeschaltet werden, da die Ausgaberoutinen unter Umständen unerwünschterweise sehr schnell große Speicherbereiche überschreiben, was ohne geeignete Systemsoftware zu nur schwer zu lokalisierbaren Fehlerquellen führt.

Wenn der Arbeitsbereich nicht begrenzt werden soll, so ist es ratsam, beim vs_clip-Aufruf die bei v_opnvwk erhaltenen Bildschirmausmaße einzustellen. Das Ausschalten des Clipping führt grundsätzlich NICHT zu einer Beschleunigung der Ausgabe.

2. Ausgabe-Funktionen

In diesem Abschnitt finden Sie die grafischen Grundfunktionen des VDI.

POLYLINE (VDI 6)

„POLYLINE“ zeichnet einen Linienzug. Alle angegebenen Punkte werden nacheinander mit Linien verbunden.
Es müssen mindestens zwei Koordinatenpaare übergeben werden.

Deklaration: void v_pline(int handle, int count, int *pxyarray);
Aufruf: v_pline(handle, count, pxyarray);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	6	v_pline
contrl[1]	n	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
ptsin[0..2n-1]	pxyarray[0..2n-1]	Koordinaten

OUTPUT BEZIER (VDI 6, 13)

Diese Funktion zeichnet eine ungefüllte Bezierkurve.

Deklaration: void v_bez(int handle, int count, int *xyarr, char *bezarr,
int *extent,int *totpts, int *totmoves);
Aufruf: v_bez(handle, count, xyarr, bezarr, extent, totpts, totmoves);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	6	v_bez
contrl[1]	n	Einträge in ptsin
contrl[2]	2	Einträge in ptsout
contrl[3]	(n+1)/2	Einträge in intin
contrl[4]	6	Einträge in intout

contrl[5]	13	signalisiert v_bez
contrl[6]	handle	
ptsin[0..2n-1]	xyarr[0..2n-1]	Koordinaten
intin[0..(n+1)/2-1]	bezarr[0..n-1]	Punkttypen
intout[0]	totpts	Anzahl der berechneten Punkte
intout[1]	totmoves	Anzahl der Unterbrechungen im Linienzug
intout[2..5]	reserviert	
ptsout[0..3]	extent[0..3]	Koordinaten des umschließenden Rechtecks

Bedeutung der Punkttypen:

Bit 0: Startpunkt eines 4-Punkte Beziersegments (2 Ankerpunkte und zwei Richtungspunkte). Der Endpunkt eines Beziersegments kann auch der Startpunkt des nächsten Beziers sein - er kann aber kein „jump point" sein.
Bit 1: „jump point". Dieser Punkt und der vorhergehende werden nicht verbunden. Nützlich um Enklaven oder Exklaven zu zeichnen.
Bit 2-7 sind undefiniert. Ist im Punkttyp Bit 0 gelöscht, verhält sich die Bezierfunktion wie eine Polyline-Funktion mit der Erweiterung, über den „jump point" Enklaven oder Exklaven zeichnen zu können.

Bemerkung:

Die im Byte-Array bezarr übergebenen Punkttypen werden vom Binding im Intel-Format im intin[] abgelegt (d.h. bezarr[0] liegt im Low-Byte von intin[0], bezarr[1] im High-Byte von intin[0]), um Kompatibilität zum PC-GEM zu gewährleisten.

POLYMARKER (VDI 7)

Diese Funktion zeichnet Marker an den angegebenen Stellen.

Deklaration: void v_pmarker(int handle, int count, int *pxyarray);
Aufruf: v_pmarker(handle, count, pxyarray);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	7	v_pmarker
contrl[1]	n	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
ptsin[0..2n-1]	pxyarray[0..2n-1]	Koordinaten

TEXT (VDI 8)

„TEXT“ gibt eine Zeichenkette mit Attributen aus.

Deklaration: void v_gtext(int handle, int x, int y, char *string);
Aufruf: v_gtext(handle, x, y, string);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	8	v_gtext
contrl[1]	1	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	n	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
intin[0..n-1]	string[0..n-1]	Zeichenkette
ptsin[0] x		
ptsin[1] y		

FILLED AREA (VDI 9)

Durch „**FILLED AREA**“ wird eine beliebige, gefüllte Fläche gezeichnet.

Deklaration: void v_fillarea(int handle, int count, int *pxyarray);
Aufruf: v_fillarea(handle, count, pxyarray);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	9	v_fillarea
contrl[1]	n	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
ptsin[0..2n-1]	pxyarray[0..2n-1]	Koordinaten

Bemerkung:

Im ATARI-VDI wird bei ausgeschalteter Umrandung bei der Ausgabe eines Rechtecks die Y1-Koordinate erhöht und die Y2-Koordinate verringert, so daß das Rechteck insgesamt um 2 Zeilen kleiner wird.

Dieser Fehler wird vom NVDI aus Kompatibilitätsgründen nachgebildet. Da ihn nicht alle Treiber nachbilden, bietet es sich zum Zeichnen eines Rechtecks an, die Funktion „**BAR**“ aufzurufen, was ohnehin schneller ist.

OUTPUT FILLED BEZIER (VDI 9, 13)

Diese Funktion zeichnet eine gefüllte Bezierkurve.

Deklaration: void v_bez_fill(int handle, int count, int *xyarr, char *bezarr, int *extent,int *totpts, int *totmoves);

Aufruf: v_bez_fill(handle, count, xyarr, bezarr, extent, totpts,totmoves);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	9	v_bez_fill
contrl[1]	n	Einträge in ptsin
contrl[2]	2	Einträge in ptsout
contrl[3]	(n+1)/2	Einträge in intin
contrl[4]	6	Einträge in intout
contrl[5]	13	signalisiert v_bez_fill
contrl[6]	handle	
ptsin[0..2n-1]	xyarr[0..2n-1]	Koordinaten
intin[0..(n+1)/2-1]	bezarr[0..n-1]	Punkttypen
intout[0]	totpts	Anzahl der berechneten Punkte
intout[1]	totmoves	Anzahl der Unterbrechungen im Linienzug
intout[2..5]	reserviert	
ptsout[0..3]	extent[0..3]	Koordinaten des umschließenden Rechtecks

Bedeutung der Punkttypen:

Bit 0: Startpunkt eines 4-Punkte Beziersegments (2 Ankerpunkte und zwei Richtungspunkte). Der Endpunkt eines Beziersegments kann auch der Startpunkt des nächsten Beziers sein - er kann aber kein „jump point“ sein.

Bit 1: „jump point“. Dieser Punkt und der vorhergehende werden nicht verbunden. Nützlich um Enklaven oder Exklaven zu zeichnen.

Bit 2-7 sind undefiniert. Ist im Punkttyp Bit 0 gelöscht, verhält sich die Bezierfunktion wie eine „FILLED AREA“-Funktion mit der Erweiterung über den „jump point“ Enklaven oder Exklaven zeichnen zu können.

Bemerkung:

Die im Byte-Array bezarr übergebenen Punkttypen werden vom Binding im Intel-Format im intin[] abgelegt (d.h. bezarr[0] liegt im Low-Byte von intin[0], bezarr[1] im High-Byte von intin[0]), um Kompatibilität zum PC-GEM zu gewährleisten.

CONTOUR FILL (VDI 103)

Diese Funktion füllt vom Startpunkt aus eine Fläche, wobei diese Fläche durch den Bildrand oder eine andere Farbe begrenzt wird. Von vielen (Drucker-) Treibern wird „**CONTOUR FILL**“ nicht unterstützt.

Deklaration: void v_contourfill(int handle, int x, int y, int index);
Aufruf: v_contourfill(handle, x, y index);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	103	v_contourfill
contrl[1]	1	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
intin[0]	index	Farbindex
ptsin[0]	x	
ptsin[1]	y	

FILL RECTANGLE (VDI 114)

„**FILL RECTANGLE**“ zeichnet ein ausgefülltes Rechteck ohne Umrahmung.

Deklaration: void vr_recfl(int handle, int *pxyarray);
Aufruf: vr_recfl(handle, pxyarray);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	114	vr_recfl
contrl[1]	2	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
ptsin[0..3]	pxyarray[0..3]	Koordinaten

GENRALIZED DRAWING PRIMITIVE (VDI 11)

BAR (VDI 11, GDP 1)

Von dieser Funktion wird ein ausgefülltes Rechteck gezeichnet. Im Gegensatz zu „**FILL RECTANGLE**" wird eine Umrahmung ausgegeben.

Deklaration: void v_bar(int handle, int *pxyarray);
Aufruf: v_bar(handle, pxyarray);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	11	GDP
contrl[1]	2	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	1	v_bar
contrl[6]	handle	
ptsin[0..3]	pxyarray[0..3]	Koordinaten

ARC (VDI 11, GDP 2)

„**ARC**" zeichnet einen Kreisbogen, dessen Start- und Endwinkel in 1/10 Grad von 0 bis 3600 angegeben werden.

Deklaration: void v_arc(int handle, int x, int y, int radius,int begang, int endang);
Aufruf: v_arc(handle, x, y, radius, begang, endang);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	11	GDP
contrl[1]	4	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	2	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	2	v_arc
contrl[6]	handle	

intin[0]	begang	Startwinkel
intin[1]	endang	Endwinkel
ptsin[0]	x	
ptsin[1]	y	
ptsin[6]	radius	Radius

PIE (VDI 11, GDP 3)

Diese Funktion zeichnet einen Kreisflächenausschnitt. Die Winkel werden in 1/10 Grad von 0 bis 3600 angegeben.

Deklaration: void v_pieslice(int handle, int x, int y, int radius,int begang, int endang);
Aufruf: v_pieslice(handle, x, y, radius, begang, endang);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	11	GDP
contrl[1]	4	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	2	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	3	v_pieslice
contrl[6]	andle	
intin[0]	begang	Startwinkel
intin[1]	endang	Endwinkel
ptsin[0]	x	
ptsin[1]	y	
ptsin[6]	radius	Radius

CIRCLE (VDI 11, GDP 4)

Die Funktion „**CIRCLE**“ zeichnet eine Kreisfläche.

Deklaration: void v_circle(int handle, int x, int y, int radius);
Aufruf: v_circle(handle, x, y, radius);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	11	GDP
contrl[1]	3	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	2	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	4	v_circle
contrl[6]	handle	
intin[0] b	egang	Startwinkel
intin[1]	endang	Endwinkel
ptsin[0]	x	
ptsin[1]	y	
ptsin[4]	radius	Radius

ELLIPSE (VDI 11, GDP 5)

Diese Funktion zeichnet eine Ellipsenfläche.

Deklaration: void v_ellipse(int handle, int x, int y, int xradius,int yradius);
Aufruf: v_ellipse(handle, x, y, xradius, yradius);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	11	GDP
contrl[1]	2	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	5	v_ellipse
contrl[6]	handle	
ptsin[0]	x	
ptsin[1]	y	
ptsin[2]	xradius	Radius in horizontaler Richtung
ptsin[3]	yradius	Radius in vertikaler Richtung

ELLIPTICAL ARC (VDI 11, GDP 6)

„**ELLIPTICAL ARC**" zeichnet einen Ellipsenbogenausschnitt. Die Winkelangabe erfolgt in 1/10 Grad von 0 bis 3600.

Deklaration: void v_ellarc(int handle, int x, int y, int xradius,int yradius, int begang, int endang);
Aufruf: v_ellarc(handle, x, y, xradius, yradius, begang, endang);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	11	GDP
contrl[1]	2	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	2	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	6	v_ellarc
contrl[6]	handle	
intin[0]	begang	Startwinkel
intin[1]	endang	Endwinkel
ptsin[0]	x	
ptsin[1]	y	
ptsin[2]	xradius	Radius in horizontaler Richtung
ptsin[3]	yradius	Radius in vertikaler Richtung

ELLIPTICAL PIE (VDI 11, GDP 7)

Die Funktion „**ELLIPTICAL PIE**" zeichnet einen Ellipsenflächenausschnitt. Die Angabe der Winkel geschieht in Zehntelgrad von 0 bis 3600.

Deklaration: void v_ellpie(int handle, int x, int y, int xradius,int yradius, int begang, int endang);
Aufruf: v_ellpie(handle, x, y, xradius, yradius, begang, endang);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	11	GDP
contrl[1]	2	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	2	Einträge in intin
contrl[4]	0	Einträge in intout

contrl[5]	7	v_ellpie
contrl[6]	handle	
intin[0]	begang	Startwinkel
intin[1]	endang	Endwinkel
ptsin[0]	x	
ptsin[1]	y	
ptsin[2]	xradius	Radius in horizontaler Richtung
ptsin[3]	yradius	Radius in vertikaler Richtung

ROUNDED RECTANGLE (VDI 11, GDP 8)

Ein Rechteck mit gerundeten Ecken wird gezeichnet.

Deklaration: void v_rbox (int handle, int *pxyarray);
Aufruf: v_rbox(handle, pxyarray);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	11	GDP
contrl[1]	2	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	8	v_rbox
contrl[6]	handle	
ptsin[0..3]	pxyarray[0..3]	Koordinaten

FILLED ROUNDED RECTANGLE (VDI 11, GDP 9)

Diese Funktion zeichnet ein ausgefülltes, abgerundetes Reckteck.

Deklaration: void v_rfbox (int handle, int *pxyarray);
Aufruf: v_rfbox(handle, pxyarray);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	11	GDP
contrl[1]	2	Einträge in ptsin

contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	9	v_rfbox
contrl[6]	handle	
ptsin[0..3]	pxyarray[0..3]	Koordinaten

Bemerkung:

Im ATARI-VDI ist bei ausgeschalteter Umrahmung die Höhe um 2 Pixel kleiner als angegeben (das beruht auf einem Fehler in v_fillarea). Dieser Fehler wird von NVDI nicht unterstützt.

JUSTIFIED GRAPHICS TEXT (VDI 11, GDP 10)

„**JUSTIFIED GRAPHICS TEXT**" ermöglicht die Ausgabe einer Zeichenkette mit Attributen und Dehnung oder Stauchung auf die gewünschte Länge.
Es können entweder Wort- oder Zeichenzwischenräume gedehnt werden.

Deklaration: void v_justified(int handle,int x, int y, char *string,int length, int word_space, int char_space);

Aufruf: v_justified(handle, x, y, string, length, word_space,char_space);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	11	GD
contrl[1]	2	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	n+2	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	10	v_justified
contrl[6]	handle	
intin[0]	word_space	<>0: Wortzwischenräume dehnen
intin[1]	char_space	<> 0: Zeichenzwischenräume dehnen
intin[2..n+1]	string[0..n-1]	Zeichenkette
ptsin[0]	x	
ptsin[1]	y	
ptsin[2]	length	horizontale Textlänge in Pixeln

Bemerkung:

Das ATARI-VDI zerstört contrl[3] - ein Unding, da Gerätetreiber die Eingabeparameter nicht verändern dürfen.

ENABLE BEZIER CAPABILITIES (VDI 11, GDP 13)

Diese Funktion schaltet die Bezierfunktionen ein.

Deklaration: int v_bez_on(int handle);
Aufruf: retval = v_bez_on(handle);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	11	GDP
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin - signalisiert v_bez_on
contrl[4]	0	Einträge in intout
contrl[5]	13	v_bez_on
contrl[6]	handle	
intout[0]	retval	Beziertiefe

Bedeutung von retval:

<retval> kann einen Wert von 0 bis 7 annehmen, der ein Maß für die Kurvenqualität darstellt.
0: minimale Qualität
7: maximale Qualität

Bemerkungen;

Da die Beziergenerierung je nach GDOS unterschiedlich sein kann, sind Rückschlüsse auf die Linienanzahl, wie sie z.B. in der GEM/3-Doku gemacht werden, unzuverlässig. Sinnvoller ist es, die Qualität der Bezierkurve aus dem Rückgabewert der Funktion *v_bez_qual()* zu ermitteln und den Rückgabewert von *v_bez_on()* nur zu benutzen, um eine vorhandene Bezierunterstützung zu ermitteln (s. Programmiertips).

Unter PC-GEM oder anderen bekannten GDOSsen mit Bezierunterstützung schaltet diese Funktion die Bezierfunktionen für alle Gerätetreiber ein (d.h. sämtliche „**POLYLINE**"-Aufrufe werden als „**BEZIER**"-Aufrufe, sämtliche „**FILLED AREA**"-Aufrufe als „**FILLED BEZIER**"-Aufrufe behandelt). Dieser Design-Fehler kann im Zusammenhang mit Accessories, spätestens jedoch im Multitasking-Betrieb zu Problemen führen.
Um diese Probleme zu vermeiden, sollten Sie diese Funktion mit einem korrespondiererden *v_bez_off()* um jeden Bezieraufruf schachteln (und beten, daß zwischendurch kein Taskswitch auftritt) oder sie verlassen sich auf ein undokumentiertes Feature:
Alle derzeit bekannten GDOSse mit Bezier-Unterstützung (auch die PC-GEM GDOSse) können die Bezierfunktionen *v_bez()* und *v_bez_fill()* von den Funktionen *v_pline()* und *v_fillarea()* durch die Parameter contrl[3] und contrl[5] unterscheiden, so daß man auf den Aufruf von *v_bez_on()* und *v_bez_off()* vollkommen verzichten kann.
NVDI ist ebenfalls in der Lage die Bezierfunktionen anhand von contrl[3] und contrl[5]

eindeutig zu identifizieren und vermeidet die oben geschilderten Probleme mit *v_bez_on()/ v_bez_off()* dadurch, daß diese Funktionen nur die mit <*handle*> angesprochene (virtuelle) Workstation beeinflussen.

DISABLE BEZIER CAPABILITIES (VDI 11, GDP 13)

Diese Funktion schaltet die Bezierfunktionen aus.

Deklaration: void v_bez_off(int handle);
Aufruf: v_bez_off(handle);

PARAMETER:

Variable	Belegung	Bedeutung
contrl[0]	11	GDP
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin - signalisiert v_bez_off
contrl[4]	0	Einträge in intout
contrl[5]	13	v_bez_off
contrl[6]	handle	

3. Attribut-Funktionen

Mit den folgenden Funktionen können Füll-, Linien-, Marker-, Text- und Verknüpfungsattribute eingestellt werden.

SET WRITING MODE (VDI 32)

Diese Funktion wählt die Verknüpfung der Grafikoperationen aus. Bei Übergabe eines nicht vorhandenen Modus wird Modus 1 (REPLACE) angewählt.

Deklaration: int vswr_mode(int handle, int mode);
Aufruf: set_mode = vswr_mode(handle, mode);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	32	vswr_mode
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[6]	handle	
intin[0]	mode	gewünschter Verknüpfungsmodus
intout[0]	set_mode	ausgewählter Verknüpfungsmodus

Bedeutung von mode:

1: **(REPLACE)** Alles, was sich unter dem Grafik-Element befindet wird überdeckt.
2: **(TRANSPARENT)** Nur die gesetzen Pixel des Grafik-Elementes überdecken den Hintergrund.
3: **(XOR bzw. EOR)** Wenn das Pixel des Grafik-Elementes und das darunterliegende Pixel die gleiche Farbe haben, so erscheint kein Punkt. Andernfalls wird der Punkt gesetzt.
4: **(REV. TRANSPARENT)** Die nicht gesetzten Pixel des Grafik-Elementes überdecken den Hintergrund.

SET COLOR REPRESENTATION (VDI 14)

Mit dieser Funktion kann man die Farbabstufung einer Farbnummer festlegen. Die Intensität von Rot, Grün und Blau wird jeweils in Promille (0-1000) angegeben.

Deklaration: void vs_color(int handle, int index, int *rgb_in);
Aufruf: vs_color(handle, index, rgb_in);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	14	vs_color
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	4	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
intin[0]	index	Farbnummer
intin[1..3]	rgb_in[0..2]	Farbintensitäten von Rot, Grün, Blau

Bemerkung:
Diese Funktion kann nur sinnvolleingesetzt werden, wenn eine „Color lookup-table"-Unterstützung vorhanden ist (was man über *vq_extnd()* erfragen kann).

SET POLYLINE LINE TYPE (VDI 15)

Mit „**SET POLYLINE LINE TYPE**" kann man den Linientyp festlegen. Wenn der gewünschte Linientyp nicht einstellbar ist, so wird der Linientyp 1 (durchgehende Linie) eingestellt.

Deklaration: int vsl_type(int handle, int style);
Aufruf: set_type = vsl_type(handle, style);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	15	vsl_type
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[6]	handle	
intin[0]	style	gewünschter Linientyp
intout[0]	set_type	ausgewählter Linientyp

Bedeutung von style:

1: %1111111111111111 (durchgehende Linie)
2: %1111111111110000 (langer Strich)
3: %1110000011100000 (Punkte)
4: %1111111100011000 (Strich, Punkt)
5: %1111111100000000 (Strich)
6: %1111000110011000 (Strich, Punkt, Punkt)
7: benutzerdefiniert (vsl_udsty)

SET USER-DEFINED LINE STYLE PATTERN (VDI 113)

Mit dieser Funktion legt man den benutzerdefinierten Linientyp von „**SET POLYLINE LINE TYPE**" fest.
Das höchstwertige Bit des Linienmusters ist der erste Punkt der Linie.

Deklaration: void vsl_udsty(int handle, int pattern);
Aufruf: vsl_udsty(handle, pattern);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	113	vsl_udsty
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
intin[0]	pattern	benutzerdefiniertes Linienmuster

SET POLYLINE LINE WIDTH (VDI 16)

Diese Funktion setzt die Linienbreite, wobei nur ungerade Werte eingestellt werden (ggf. wird auf den nächstkleineren Wert gerundet).

Deklaration: int vsl_width(int handle, int width);
Aufruf: set_width = vsl_width(handle, width);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	16	vsl_width
contrl[1]	1	Einträge in ptsin
contrl[2]	1	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
ptsin[0]	width	gewünschte Linienbreite
ptsout[0]	set_width	ausgewählte Linienbreite

Bemerkung:

Die Linienbreite orientiert sich immer an der horitontalen Pixelgröße.

SET POLYLINE COLOR INDEX (VDI 17)

Die Linienfarbe wird gesetzt. Bei ungültigem Farbindex wird der Farbindex 1 gesetzt.

Deklaration: int vsl_color(int handle, int color_index);
Aufruf: set_color = vsl_color(handle, color_index);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	17	vsl_color
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[6]	handle	
intin[0]	color_index	gewünschte Linienfarbe
intout[0]	set_color	ausgewählte Linienfarbe

SET POLYLINE END STYLES (VDI 108)

Das Aussehen der Linienenden wird mit „**SET POLYLINE END STYLES**“ bestimmt. Bei ungültigen Angaben wird das betreffende Linienende eckig.

Deklaration: void vsl_ends(int handle, int beg_style, int end_style);
Aufruf: vsl_ends(handle, beg_style, end_style);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	108	vsl_ends
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	2	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
intin[0]	beg_style	Aussehen des Linienanfangs
intin[1]	end_style	Aussehen des Linienendes

Bedeutung von beg_style und end_style:

0: eckig
1: Pfeil
2: abgerundet

SET POLYMARKER TYPE (VDI 18)

Mit dieser Funktion wird der gewünschte Marker ausgewählt. Im Fall einer fehlerhaften Markernummer wird Markertyp 3 benutzt.

Deklaration: int vsm_type(int handle, int symbol);
Aufruf: set_type = vsm_type(handle, symbol);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	18	vsm_type
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin

contrl[4]	1	Einträge in intout
contrl[6]	handle	
intin[0]	symbol	gewünschter Markertyp
intout[0]	set_type	ausgewählter Markertyp

Bedeutung von symbol:

1: Punkt
2: Plus
3: Sternchen
4: Quadrat
5: Kreuz
6: Raute

SET POLYMARKER HEIGHT (VDI 19)

Die Markergröße kann mittels „**SET POLYMARKER HEIGHT**" eingestellt werden. Falls die eingestellte Höhe nicht existiert, wird die nächstkleinere Höhe eingestellt.

Deklaration: int vsm_height(int handle, int height);
Aufruf: set_height = vsm_height(handle, height);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	19	vsm_height
contrl[1]	1	Einträge in ptsin
contrl[2]	1	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
ptsin[1]	height	gewünschte Markerhöhe
ptsout[0]	set_width	ausgewählte Markerbreite
ptsout[1]	set_height	ausgewählte Markerhöhe

SET POLYMARKER COLOR INDEX (VDI 20)

Diese Funktion setzt die Farbe der Marker. Bei ungültigem Farbindex wird der Farbindex 1 gesetzt.

Deklaration: int vsm_color(int handle, int color_index);
Aufruf: set_color = vsm_color(handle, color_index);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	20	vsm_color
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[6]	handle	
intin[0]	color_index	gewünschte Markerfarbe
intout[0]	set_color	ausgewählte Markerfarbe

SET CHARACTER HEIGHT, ABSOLUTE MODE (VDI 12)

Die Zeichenhöhe von der Basislinie bis zur Zeichenzellenobergrenze kann mit dieser Funktion festgelegt werden. Sofern kein Font in der gewünschten Höhe vorhanden ist, wird auf die gewünschte Höhe vergrößert oder verkleinert.

Deklaration: void vst_height(int handle, int height, int *char_width,int *char_height, int *cell_width, int *cell_height);

Aufruf: vst_height(handle, height, &char_width, &char_height,&cell_width, &cell_height);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	12	vst_height
contrl[1]	1	Einträge in ptsin
contrl[2]	2	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
ptsin[1]	height	gewünschte Zeichenhöhe
ptsout[0]	char_width	ausgewählte Zeichenbreite
ptsout[1]	char_height	ausgewählte Zeichenhöhe
ptsout[2]	cell_width	ausgewählte Zeichenzellenbreite
ptsout[3]	cell_height	ausgewählte Zeichenzellenhöhe

SET CHARACTER HEIGHT, POINTS MODE (VDI 107)

Mit „**SET CHARACTER HEIGHT, POINTS MODE**“ kann die Zeichenzellengröße in Punkten (1/72") festgelegt werden.
Diese Funktion sucht den Zeichensatz heraus, der in einfacher oder doppelter Vergrößerung kleiner oder gleich der gewünschten Höhe ist (den Satz gleich noch mal durchlesen!).

Beispiel:

Der Systemzeichensatz liegt in Höhen von 8, 9 und 10 Punkten vor. Wenn man nun vst_point eine Höhe von 19 Punkten übergibt, so wird vom Gerätetreiber eine Höhe von 18 Punkten eingestellt und der 9-Punkt-Font wird in doppelter Vergrößerung ausgegeben. Wird *vst_point* eine Höhe von 7 Punkten übergeben, so wird der 8-Punkt-Font selektiert, da er der kleinste vorhandene Font ist.

Deklaration: int vst_point(int handle, int point,int *char_width, int *char_height,int *cell_width, int *cell_height);
Aufruf: set_point = vst_point(handle, point, &char_width,&char_height, &cell_width,&cell_height);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	107	vst_point
contrl[1]	0	Einträge in ptsin
contrl[2]	2	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[6]	handle	
intin[0]	point	gewünschte Zeichenzellenhöhe (1/72")
intout[0]	set_point	ausgewählte Zeichenzellenhöhe (1/72")
ptsout[0]	char_width	ausgewählte Zeichenbreite
ptsout[1]	char_height	ausgewählte Zeichenhöhe
ptsout[2]	cell_width	ausgewählte Zeichenzellenbreite
ptsout[3]	cell_height	ausgewählte Zeichenzellenhöhe

SET CHARACTER BASELINE VECTOR (VDI 13)

Mit dieser Funktion kann man die Textdrehung in 1/10 Grad einstellen. Die meisten Gerätetreiber (auch die Bildschirmtreiber des NVDI) erlauben jedoch nur eine Rotation in 90 Grad-Schritten.

Deklaration: int vst_rotation(int handle, int angle);
Aufruf: set_baseline = vst_rotation(handle, angle);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	13	vst_rotation
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[6]	handle	
intin[0]	angle	gewünschter Rotationswinkel
intout[0]	set_baseline	ausgewählter Rotationswinkel

SET TEXT FACE (VDI 21)

Diese Funktion wählt - sofern vorhanden - einen Zeichensatz aus.

Deklaration: int vst_font(int handle, int font);
Aufruf: set_font = vst_font(handle, font);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	21	vst_font
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[6]	handle	
intin[0]	font	gewünschter Zeichensatz
intout[0]	set_font	ausgewählter Zeichensatz

SET GRAPHIC TEXT COLOR INDEX (VDI 22)

Diese Funktion setzt die Farbe des Textes. Bei ungültigem Farbindex wird der Farbindex 1 gesetzt.

Deklaration: int vst_color(int handle, int color_index);
Aufruf: set_color = vst_color(handle, color_index);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	22	vst_color
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[6]	handle	
intin[0]	color_index	gewünschte Textfarbe
intout[0]	set_color	ausgewählte Textfarbe

SET GRAPHIC TEXT SPECIAL EFFECTS (VDI 106)

Mit „**SET GRAPHIC TEXT SPECIAL EFFECTS**" kann man, wie es der Name schon andeutet, spezielle Texteffekte einstellen.

Deklaration: int vst_effects(int handle, int effect);
Aufruf: set_effect = vst_effects(handle, effect);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	106	vst_effects
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[6]	handle	
intin[0]	effect	gewünschter Texteffekt
intout[0]	set_effect	ausgewählter Texteffekt

Bedeutung von effect (Bitnummer):

0: fett	**fett**
1: hell	hell
2: kursiv	*kursiv*
3: unterstrichen	unterstrichen
4: umrandet	umrandet

SET GRAPHIC TEXT ALIGNMENT (VDI 39)

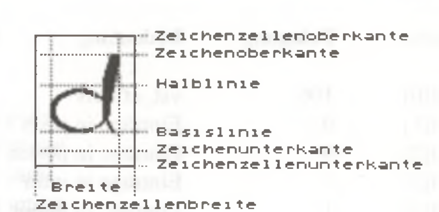
Die horizontale und vertikale Ausrichtung eines Textes kann mit dieser Funktion beeinflußt werden. Bei falscher Eingabe für horizontale Ausrichtung wird der Text linksjustiert. Die fehlerhafte Angabe der vertikalen Ausrichtung bewirkt Ausrichtung an der Basislinie.

Deklaration: void vst_alignment(int handle, int hor_in, int vert_in,int *hor_out, int *vert_out);
Aufruf: vst_alignment(handle, hor_in, vert_in, &hor_out, &vert_out);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	39	vst_alignment
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	2	Einträge in intin
contrl[4]	2	Einträge in intout
contrl[6]	handle	
intin[0]	hor_in	gewünschte horizontale Ausrichtung
intin[1]	vert_in	gewünschte vertikale Ausrichtung
intout[0]	hor_out	ausgewählte horizontale Ausrichtung
intout[1]	vert_out	ausgewählte vertikale Ausrichtung

Bedeutung von hor_in:
0: linksjustiert
1: zentriert
2: rechtsjustiert



Bedeutung von vert_in:
0: Basislinie
1: Halblinie
2: Zeichenoberkante
3: Zeichenzellenunterkante
4: Zeichenunterkante
5: Zeichenzellenoberkante

SET FILL INTERIOR INDEX (VDI 23)

Der Fülltyp kann mit dieser Funktion ausgewählt werden. Bei Übergabe eines ungültigem Fülltyps wird der Typ 0 (leer) eingestellt.

Deklaration: int vsf_interior(int handle, int style);
Aufruf: set_interior = vsf_interior(handle, style);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	23	vsf_interior
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[6]	handle	
intin[0]	style	gewünschter Fülltyp
intout[0]	set_interior	ausgewählter Fülltyp

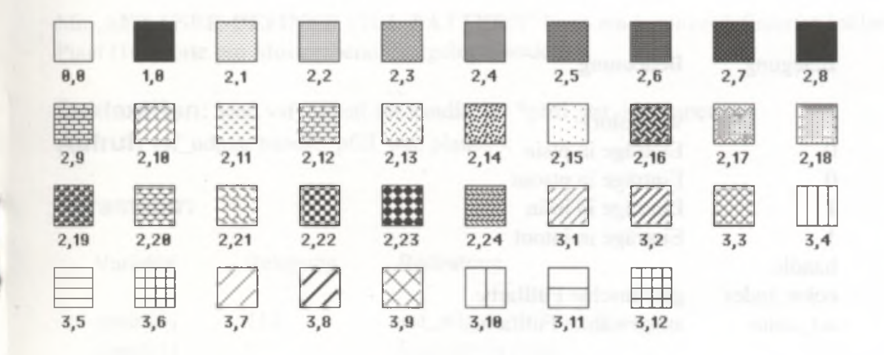
Bedeutung von style:

0: leer
1: voll
2: gemustert
3: schraffiert
4: benutzerdefiniert

SET FILL STYLE INDEX (VDI 24)

Mit dieser Funktion wird der zum Fülltyp gehörende Füllindex gesetzt.

Deklaration: int vsf_style(int handle, int style_index);
Aufruf: set_style = vsf_style(handle, style_index);



Parameter:

Variable	Belegung	Bedeutung
contrl[0]	24	vsf_style
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[6]	handle	
intin[0]	style_index	gewünschter Füllindex
intout[0]	set_sytle	ausgewählter Füllindex

Bemerkung:

Diese Funktion ist nur sinnvoll einsetzbar, wenn vorher der Fülltyp 2 (gemustert) oder 3 (schraffiert) eingestellt wurde.

SET FILL COLOR INDEX (VDI 25)

Die Füllfarbe wird anhand dieser Funktion ausgewählt. Bei ungültigem Farbindex wird der Farbindex 1 gesetzt.

Deklaration: int vsf_color(int handle, int color_index);
Aufruf: set_color = vsf_color(handle, color_index);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	25	vsf_color
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[6]	handle	
intin[0]	color_index	gewünschte Füllfarbe
intout[0]	set_color	ausgewählte Füllfarbe

SET FILL PERIMETER VISIBILITY (VDI 104)

Die Umrahmung einer gefüllten Fläche (Rechteck, Polygon, Ellipse,...) kann mit dieser Funktion ein- oder ausgeschaltet werden.

Deklaration: int vsf_perimeter(int handle, int per_vis);
Aufruf: set_perimeter = vsf_perimter(handle, per_vis);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	104	vsf_perimeter
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[6]	handle	
intin[0]	per_vis	gewünschtes Umrahmungsflag
intout[0]	set_perimeter	ausgewähltes Umrahmungsflag

Bedeutung von per_vis:
0: keine Umrahmung
1: Umrahmung

SET USER-DEFINED FILL PATTERN (VDI 112)

Mit „**SET USER-DEFINED FILL PATTERN**" kann ein benutzerdefiniertes Füllmuster von 16*16 Pixel (16 Worte pro Musterebene) festgelegt werden.

Deklaration: void vsf_udpat(int handle, int *pfill_pat, int planes);
Aufruf: vsf_udpat(handle, pfill_pat, planes);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	112	vsf_udpat
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	16n	Einträge in intin (Musterebenen*16)
contrl[4]	0	Einträge in intout
contrl[6]	handle	
intin[0..16n-1]	pfill_pat[0..16n-1]	Musterebenen

4. Rasteroperationen

Mit einem Teil dieser Funktionen können Sie Pixelblöcke pixelweise verknüpfen bzw. in ein anderes Datenformat transformieren. Hierfür werden sogenannte MFDBs (Memory Form Definition Block) benutzt.

Aufbau eines MFDB:

```
typedef struct
{
        void    *fd_addr;
        int     fd_w;
        int     fd_h;
        int     fd_wdwidth;
        int     fd_stand;
        int     fd_nplanes;
        int     fd_r1;
        int     fd_r2;
        int     fd_r3;
} MFDB;
```

fd_addr: Anfangsadresse des Rasters. Wenn hier 0 übergeben wird, so trägt der Gerätetreiber die für den Bildschirm erforderlichen Daten ein. D.h. Sie können sich das Setzen von fd_w, fd_h, fd_wdwidth und fd_nplanes ersparen.
fd_w: Breite einer Rasterzeile in Punkten
fd_h: Rasterhöhe in Punkten
fd_wdwidth: Breite einer Rasterzeile in Worten
fd_stand: Rasterformat:
0: gerätespezifisch
1: Standardformat
fd_nplanes: Anzahl der Bildebenen
fd_r1 - r3: reserviert. Diese Elemente sollten mit 0 besetzt werden!

Im Standardformat liegt jeweils eine gesamte Bildebene hinter der anderen, während im gerätespezifischen Format auf dem ST die Bildebenen wortweise hintereinander liegen.

COPY RASTER, OPAQUE (VDI 109)

„COPY RASTER, OPAQUE“ kopiert pixelweise ein rechteckiges Raster auf ein anderes rechteckiges Raster. Hierbei werden die angegebenen logischen Verknüpfungen beachtet.

Deklaration: void vro_cpyfm(int handle, int vr_mode, int *pxyarray,MFDB *psrcMFDB, MFDB *pdesMFDB);

Aufruf: vro_cpyfm(handle, wr_mode, pxyarray, &pscrMFDB, &pdesMFDB);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	109	vro_cpyfm
contrl[1]	4	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
contrl[7..8]	psrcMFDB	Zeiger auf den MFDB des Quellrasters
contrl[9..10]	pdesMFDB	Zeiger auf den MFDB des Zielrasters
intin[0]	wr_mode	logische Verknüpfung
ptsin[0..7]	pxyarray[0..7]	Koordinaten

Bedeutung von wr_mode (logische Verknüpfungen):

0: Ergebnis=0
1: Ergebnis=Quelle and Ziel
2: Ergebnis=Quelle and (not Ziel)
3: Ergebnis=Quelle
4: Ergebnis=(not Quelle) and Ziel
5: Ergebnis=Ziel (sinnlos!)
6: Ergebnis=Quelle xor Ziel
7: Ergebnis=Quelle or Ziel
8: Ergebnis=not (Quelle or Ziel)
9: Ergebnis=not (Quelle xor Ziel)
10: Ergebnis=not Ziel
11: Ergebnis=Quelle or (not Ziel)
12: Ergebnis=not Quelle
13: Ergebnis=(not Quelle) or Ziel
14: Ergebnis=not (Quelle and Ziel)
15: Ergebnis=1

Bedeutung von pxyarray:

pxyarray[0..3]: Koordinaten des Quellrechtecks
pxyarray[4..7]: Koordinaten des Zielrechtecks

Bemerkung:

Wenn der Bildschirm Quelle oder Ziel einer Rasteroperation ist, bietet es sich an, *fd_addr* IMMER auf 0 zu setzen, da nur dann ein Clipping erfolgen kann und da nur dann mit (Farb) Großbildschirmen keine Probleme zu erwarten sind!

COPY RASTER, TRANSPARENT (VDI 121)

Diese Funktion überträgt unter Berücksichtigung von Vorder- und Hintergrundfarbe sowie des Schreibmodus ein zweifarbiges (aus einer Ebene bestehendes) Quellraster zu einem mehrfarbigen Zielraster.

Deklaration: void vrt_cpyfm(int handle, int vr_mode, int *pxyarray,MFDB *psrcMFDB, MFDB *pdesMFDB,int *color_index);

Aufruf: vrt_cpyfm(handle, vr_mode, pxyarray, &psrcMFDB, &pdesMFDB);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	121	vrt_cpyfm
contrl[1]	4	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	3	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
contrl[7..8]	psrcMFDB	Zeiger auf den MFDB des Quellrasters
contrl[9..10]	pdesMFDB	Zeiger auf den MFDB des Zielrasters
intin[0]	wr_mode	Schreibmodus
intin[1]	color_index[0]	Farbindex der gesetzten Punkte
intin[2]	color_index[1]	Farbindex der gelöschten Punkte
ptsin[0..7]	pxyarray[0..7]	Koordinaten

Bedeutung von pxyarray:

pxyarray[0..3]: Koordinaten des Quellrechtecks
pxyarray[4..7]: Koordinaten des Zielrechtecks

Bemerkung:

Ebenso wie bei *vro_cpyfm* sollte auch bei *vrt_cpyfm fd_addr* 0 sein, wenn der Bildschirm Ziel einer Rasteroperation ist!

TRANSFORM FORM (VDI 110)

Diese Funktion transformiert ein Raster vom Standardformat ins gerätespezifische Format und umgekehrt.

Deklaration: void vr_trnfm(int handle, MFDB *psrcMFDB, MFDB *pdesMFDB);

Aufruf: vr_trnfm(handle, &psrcMFDB, &pdesMFDB);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	110	vr_trnfm
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
contrl[7...8]	psrcMFDB	Zeiger auf den MFDB des Quellrasters
contrl[9...10]	pdesMFDB	Zeiger auf den MFDB des Zielrasters

Bemerkung:

Bevor ein im Standardformat vorliegendes Raster mit vro_cpyfm oder vrt_cpyfm auf den Bildschirm kopiert wird, muß es per *vr_trnfm* ins gerätespezifische Format umgewandelt werden! Geschieht dies nicht, so kann Müll auf dem Bildschirm erscheinen.
Näheres dazu siehe Abschnitt „Programmiertips".

GET PIXEL (VDI 105)

„**GET PIXEL**" ermittelt den Zustand (gelöscht/gesetzt) und die Farbe eines Pixels.

Deklaration: void v_get_pixel(int handle, int x, int y, int *pel,int *index);
Aufruf: v_get_pixel(handle, x, y, &pel, & index);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	105	v_get_pixel
contrl[1]	1	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	2	Einträge in intout
contrl[6]	handle	
ptsin[0]	x	
ptsin[1]	y	
intout[0]	pel	Pixelwert
intout[1]	index	Farbindex des Pixels

Bemerkung:
Im HiColor- oder TrueColor-Modus kann es passieren, daß der Pixelwert nicht in der Farbpalette vorhanden ist oder nicht in einem Wort dargestellt werden kann. In diesem Fall wird als Farbindex -1 zurückgegeben.

5. Eingabefunktionen

Die nachfolgenden Funktionen erlauben es, von verschiedenen Eingabegeräten unter Berücksichtigung der Eingabenmodi (s. vsin_mode) Eingaben des Anwenders entgegenzunehmen.
Diese Funktionen sollten nur von dem Programm benutzt werden, das die physikalische Workstation geöffnet hat (beim Bildschirmtreiber i.a. also das AES). Andernfalls könnten der physikalischen Workstation Eingaben „geklaut" werden.

SET INPUT MODE (VDI 33)

Mit „**SET INPUT MODE**" kann man für ein bestimmtes Eingabegerät den Eingabemodus festlegen.

Deklaration: void vsin_mode(int handle, int dev_type, int mode);
Aufruf: vsin_mode(handle, dev_type, mode);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	33	vsin_mode
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	2	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[6]	handle	
intin[0]	dev_type	Eingabeeinheit
intin[1]	mode	gewünschter Eingabemodus
intout[0]	set_mode	ausgewählter Eingabemodus

Bedeutung von dev_type:

1: Maus
2: Cursor
3: Funktionstasten
4: Tastatur

Bedeutung von mode:

1: (REQUEST) Eingabeeinheit abfragen und sofort den Eingabewert und den Status zurückgeben.
2: (SAMPLE) Warten bis eine Eingabe erfolgt und dann den Eingabewert zurückgeben.

INPUT LOCATOR, REQUEST MODE (VDI 28)

Mit dieser Funktion wird die Position des Mauszeigers ermittelt und eine neue Position übergeben. Der Mauszeiger wird erst nach Druck einer Maustaste neu positioniert.

Deklaration: void vrq_locator(int handle, int x, int y, int *xout,int *yout, int *term);
Aufruf: vrq_locator(handle, x, y, &xout, &yout, &term);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	28	vrq_locator
contrl[1]	1	Einträge in ptsin
contrl[2]	1	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[6]	handle	
ptsin[0]	x neue	x-Koordinate des Mauszeigers
ptsin[1]	y neue	y-Koordinate des Mauszeigers
intout[0]	term	Maustastenstatus+31
ptsout[0]	xout alte	x-Koordinate des Mauszeigers
ptsout[1]	yout alte	y-Koordinate des Mauszeigers

INPUT LOCATOR, SAMPLE MODE (VDI 28)

Mit dieser Funktion wird die Position des Mauszeigers ermittelt und eine neue Position übergeben. Tastenbetätigungen oder Mausposition werden nur dann gemeldet, wenn sie wirklich erfolgt sind.

Deklaration: int vsm_locator(int handle, int x, int y, int *xout, int*yout, int *term);
Aufruf: status = vsm_locator(handle, x, y, &xout, &yout, &term);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	28	vsm_locator
contrl[1]	1	Einträge in ptsin
contrl[2]	0	oder 1 Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	oder 1 Einträge in intout
contrl[6]	handle	
ptsin[0]	x neue	x-Koordinate des Mauszeigers

ptsin[1]	y neue	y-Koordinate des Mauszeigers
intout[0]	term	Maustastenstatus+31
ptsout[0]	xout	alte x-Koordinate des Mauszeigers
ptsout[1]	yout	alte y-Koordinate des Mauszeigers

Bedeutung von status (gebildet aus (contrl[4]«1)|contrl[2]) (Bitnummer):

0: Positionsveränderung
1: Tastendruck

INPUT CHOICE, REQUEST MODE (VDI 30)

Mit dieser Funktion wird die Betätigung einer Funktionstaste abgewartet und die Tastennummer (1-10) zurückgegeben.

Deklaration: void vrq_choice(int handle, int ch_in, int *ch_out);
Aufruf: vrq_choice(handle, ch_in, &ch_out);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	30	vrq_choice
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[6]	handle	
intin[0]	ch_in	initialisierende Taste (0)
intout[0]	ch_out	ausgewählte Funktionstaste

Bemerkung:

Diese Funktion gibt im ATARI-VDI immer 1 zurück. Im NVDI ist sie funktionstüchtig.

INPUT CHOICE, SAMPLE MODE (VDI 30)

Sofern eine Funktionstaste betätigt wurde, gibt diese Funktion die Tastennummer (1-10) zurück.

Deklaration: int vsm_choice(int handle, int *choice);
Aufruf: status = vsm_choice(handle, &choice);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	30	vsm_choice
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0 oder 1	Einträge in intout
contrl[6]	handle	
intout[0]	choice	Tastennummer

Bedeutung von status (contrl[4]):

0: kein Tastendruck

1: Tastendruck erfolgt

Bemerkung:

Diese Funktion gibt im ATARI-VDI immer 1 zurück. Im NVDI funktioniert sie.

INPUT STRING,REQUEST MODE (VDI 31)

Diese Funktion gibt eine Zeichenkette von der Tastatur zurück, wenn RETURN gedrückt oder die maximale Länge erreicht wird. Abhängig von dem Flag echo_mode kann eine Ausgabe auf dem Bildschirm erfolgen.

Deklaration: void vrq_string(int handle, int max_length, int echo_mode,int *echo_xy, char *string);
Aufruf: vrq_string(handle, max_length, echo_mode, echo_xy, string);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	31	vrq_string
contrl[1]	1	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	2	Einträge in intin
contrl[4]	n	Einträge in intout
contrl[6]	handle	
intin[0]	max_length	maximale Länge der Zeichenkette
intin[1]	echo_mode	0: keine Ausgabe, 1: Ausgabe
ptsin[0]	echo_xy[0]	
ptsin[1]	echo_xy[1]	
intout[0..n-1]	string[0..n-1]	Eingabepuffer

Bedeutung von max_length:

max_length gibt die maximale Länge der Zeichenkette an. Ist max_length negativ, so wird der Absolutbetrag als Länge betrachtet, und statt der ASCII-Codes werden Scan-Codes übergeben.

INPUT STRING,SAMPLE MODE (VDI 31)

Diese Funktion gibt eine Zeichenkette von der Tastatur zurück, wenn RETURN gedrückt oder die maximale Länge erreicht wird. Sofern keine Eingaben gemacht werden, bricht die Funktion ab. Abhängig von dem Flag echo_mode kann eine Ausgabe auf dem Bildschirm erfolgen.

Deklaration: int vsm_string(int handle, int max_length, int echo_mode,int *echo_xy, char *string);
Aufruf: status = vsm_string(handle, max_length, echo_mode, echo_xy,string);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	31	vsm_string
contrl[1]	1	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	2	Einträge in intin
contrl[4]	n	Einträge in intout
contrl[6]	handle	
intin[0]	max_length	maximale Länge der Zeichenkette
intin[1]	echo_mode	0: keine Ausgabe, 1: Ausgabe
ptsin[0]	echo_xy[0]	
ptsin[1]	echo_xy[1]	
intout[0..n-1]	string[0..n-1]	Eingabepuffer

Bedeutung von max_length:

max_length gibt die maximale Länge der Zeichenkette an. Ist max_length negativ, so wird der Absolutbetrag als Länge benutzt und statt der ASCII-Codes werden Scan-Codes übergeben.

Bedeutung von status (contrl[4]):

0: keine Eingabe
<> 0: Länge der Zeichenkette

SET MOUSE FORM (VDI 111)

Das Aussehen des Mauszeigers kann mit „**SET MOUSE FORM**" frei definiert werden.

Deklaration: void vsc_form(int handle, int *pcur_form);
Aufruf: vsc_form(handle, pcur_form);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	111	vsc_form
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	37	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
intin[0..36]	pcur_form[0..36]	Mauszeigerdefinition

Bedeutung von pcur_form:

pcur_form[0]:	relative Koordinate des horizontalen Aktionspunktes
pcur_form[1]:	relative Koordinate des vertikalen Aktionspunktes
pcur_form[2]:	muß 1 sein (REPLACE)
pcur_form[3]:	Farbindex der Hintergrundmaske
pcur_form[4]:	Farbindex der Vordergrundmaske
pcur_form[5..20]:	Hintergrundmaske
pcur_form[21..36]:	Vordergrundmaske

Bemerkung:

Zum Setzen der Mausform sollte in GEM-Programmen unbedingt die AES-Funktion graf_mouse() verwendet werden. Andernfalls könnte die Mausform-Verwaltung eines Multitasking-GEM nachhaltig verwirrt werden.

Mit NVDI ist es möglich, die aktuelle Mausform zurückgeliefert zu bekommen.

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	111	vgc_form
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	37	Einträge in intout
contrl[6]	handle	
intout[0..36]	pcur_form[0..36]	Mauszeigerdefinition

EXCHANGE TIMER INTERRUPT VECTOR (VDI 118)

Mit dieser Funktion kann man eine eigene Routine im Timerinterrupt aufrufen lassen. Diese Routine muß an ihrem Ende alle veränderten Register restaurieren und die alte Timerinterruptroutine anspringen.

Deklaration: void vex_timv(int handle, long tim_addr, long *otim_addr,int *tim_conv);
Aufruf: vex_timv(handle, tim_addr, &otim_addr, &tim_conv);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	118	vex_timv
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[6]	handle	
contrl[7..8]	tim_addr	Adresse der neuen Interruptroutine
contrl[9..10]	otim_addr	Adresse der alten Interruptroutine
intout[0]	tim_conv	Interruptintervall in ms

SHOW CURSOR (VDI 122)

Mit „**SHOW CURSOR**“ hebt man einen vorhergehenden „**HIDE CURSOR**“-Aufruf auf. Wenn man den Mauszeiger sofort erscheinen lassen möchte, so muß der Parameter reset Null sein.

Deklaration: void v_show_c(int handle, int reset);
Aufruf: v_show_c(handle, reset);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	122	v_show_c
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
intin[0]	reset	

Bedeutung von reset:

0: Mauszeiger sofort anzeigen

<> 0: Hide-Counter dekrementieren und gegebenenfalls Mauszeiger zeichnen

Bemerkung:

Zum Ein-/Ausschalten der Maus sollte in GEM-Programmen unbedingt die AES-Funktion graf_mouse() verwendet werden. Andernfalls könnte die Maus-Verwaltung eines Multitasking-GEM nachhaltig verwirrt werden.

HIDE CURSOR (VDI 123)

Diese Funktion schaltet den Mauszeiger aus.

Deklaration: void v_hide_c(int handle);
Aufruf: v_hide_c(handle);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	123	v_hide_c
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	

Bemerkung:

Zum Ein-/Ausschalten der Maus sollte in GEM-Programmen unbedingt die AES-Funktion graf_mouse() verwendet werden. Andernfalls könnte die Maus-Verwaltung eines Multitasking-GEM nachhaltig verwirrt werden.

SAMPLE MOUSE BUTTON STATE (VDI 124)

Diese Funktion gibt Informationen über die Mauszeiger-Position und den Status der Maustasten zurück.

Deklaration: void vq_mouse(int handle, int *pstatus, int *x, int *y);
Aufruf: vq_mouse(handle, &pstatus, &x, &y);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	124	vq_mouse
contrl[1]	0	Einträge in ptsin
contrl[2]	1	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[6]	handle	
intout[0]	pstatus	Maustastenstatus
ptsout[0]	x	
ptsout[1]	y	

Bedeutung von pstatus:

0: keine Maustaste gedrückt
1: linke Maustaste gedrückt
2: rechte Maustaste gedrückt
3: beide Maustasten gedrückt

Bemerkung:

In GEM-Programmen sollte die AES-Funktion graf_mkstate() verwendet werden, um nur die für die eigene Applikation bestimmten Informationen über Mauszeiger-Position und Maustastenstatus zu erhalten.

EXCHANGE BUTTON CHANGE VECTOR (VDI 125)

Mit „**EXCHANGE BUTTON CHANGE VECTOR**" kann man eine Routine installieren, die beim Druck einer Maustaste aufgerufen wird und in Register d0 den Status der Maustasten enthält. Diese Routine muß alle veränderten Register wiederherstellen und die alte Maustasten-Status-Routine aufrufen.

Deklaration: void vex_butv(int handle, long pusrcode, long *psavcode);
Aufruf: vex_butv(handle, pusrcode, &psavcode);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	125	vex_butv
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
contrl[7..8]	pusrcode	Adresse der neuen Routine
contrl[9..10]	psavcode	Adresse der alten Routine

EXCHANGE MOUSE MOVEMENT VECTOR (VDI 126)

Diese Funktion erlaubt im Fall von Mausbewegungen den Aufruf einer Anwenderroutine, der in d0 und d1 die Koordinaten des Mauszeigers übergeben werden. Alle veränderten Register müssen von dieser Routine restauriert werden. Anschließend sollte die alte Mausbewegungsroutine aufgerufen werden.

Deklaration: void vex_motv(int handle, long pusrcode, long *psavcode);
Aufruf: vex_motv(handle, pusrcode, &psavcode);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	126 v	ex_motv
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
contrl[7..8]	pusrcode	Adresse der neuen Routine
contrl[9..10]	psavcode	Adresse der alten Routine

EXCHANGE CURCOR CHANGE VECTOR (VDI 127)

Mit „**EXCHANGE CURSOR CHANGE VECTOR**“ kann man eine Routine installieren, die bei Mausbewegungen aufgerufen wird. Der Aufruf dieser Anwender-Routine erfolgt, nachdem die über vex_motv eingetragene Routine aufgerufen und die Mauszeiger-Koordinaten, die man in d0 und d1 erhält, geclippt wurden. Alle veränderten Register müssen wiederhergestellt werden. Anschließend sollte die alte Routine aufgerufen werden.

Deklaration: void vex_curv(int handle, long pusrcode, long *psavcode);
Aufruf: vex_curv(handle, pusrcode, &psavcode);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	127	vex_curv
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin

contrl[4]	0	Einträge in intout
contrl[6]	handle	
contrl[7..8]	pusrcode	Adresse der neuen Routine
contrl[9..10]	psavcode	Adresse der alten Routine

Bemerkung:

Diese Funktion ist in sämtlichen bekannten Dokumentationen falsch beschrieben worden - es handelt sich nicht um eine Funktion für das Einklinken von Redraw-Routinen.

SAMPLE KEYBOARD STATE INFORMATION (VDI 128)

Diese Funktion gibt den Status der CONTROL-, ALTERNATE- sowie der SHIFT-Taste(n) zurück.

Deklaration: void vq_key_s(int handle, int *pstatus);
Aufruf: vq_key_s(handle, &pstatus);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	128	vq_key_s
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[6]	handle	
intout[0]	pstatus	Tastenstatus

Bedeutung von pstatus (Bitnummer):

0: rechte Shift-Taste
1: linke Shift-Taste
2: Control-Taste
3: Alternate-Taste

Bemerkung:

In GEM-Programmen sollten die AES Event-Funktionen verwendet werden, um nur die für die eigene Applikation bestimmten Informationen über den Tastaturstatus zu erhalten.

6. Auskunftsfunktionen

Mit diesen Funktionen kann man die aktuellen Einstellungen der VDI-Funktionen erfragen. Durch Verwendung dieser Funktionen erübrigt sich der Zugriff auf Line-A-Variablen.

EXTENDED INQUIRE FUNCTION (VDI 102)

Von dieser Funktion werden entweder die Parameter von v_opnwk/v_opnvwk oder erweiterte Auskünfte zum Gerätetreiber und zum Gerät zurückgeliefert.

Deklaration: void vq_extnd(int handle, int owflag, int *work_out);
Aufruf: vq_extnd(handle, owflag, work_out);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	102	vq_extnd
contrl[1]	0	Einträge in ptsin
contrl[2]	6	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	45	Einträge in intout
contrl[6]	handle	
intin[0]	owflag	Informationstyp
intout[0..44]	work_out[0..44]	
ptsout[0..11]	work_out[45..56]	

Bedeutung von owflag:

0: Parameter von v_opnwk/v_opnvwk
1: erweiterte Parameter

Bedeutung von work_out:

work_out[0]: Bildschirmtyp
- 0: kein Bildschirm
- 1: getrennter Text- und Grafikmodus und getrennter Bildspeicher
- 2: getrennter Text- und Grafikmodus mit gemeinsamem Bildspeicher
- 3: gemeinsamer Text- und Grafikmodus mit getrenntem Bildspeicher
- 4: gemeinsamer Text- und Grafikmodus mit gemeinsamem Bildspeicher

work_out[1]: Anzahl der Farbabstufungen
work_out[2]: Anzahl der Texteffekte
work_out[3]: Flag für Vergrößerung des Rasters
- 0: Vergrößerung nicht möglich
- 1: Vergrößerung möglich

work_out[4]: Anzahl der Bildebenen
work_out[5]: „Color lookup table"-Unterstützung
0: nicht möglich
1: möglich
work_out[6]: Anzahl der 16*16-Pixel-Raster-Operationen pro Sekunde
work_out[7]: Verfügbarkeit der Flächenfüllung (v_contourfill)
0: nicht verfügbar
1: verfügbar
work_out[8]: Textrotation
0: nicht möglich
1: in 90-Grad-Schritten
2: in 1/10-Grad-Schritten
work_out[9]: Anzahl der Schreibmodi
work_out[10]: Eingabemodi
0: keine
1: Request
2: Request und Sample
work_out[11]: Textausrichtung:
0: nicht verfügbar
1: verfügbar
work_out[12]: Farbstiftwechsel
0: nicht möglich
1: möglich
work_out[13]: Farbbandwechsel
0: nicht möglich
1: farbige Zeilen
2: farbige Zeilen und Rechtecke
work_out[14]: maximale Anzahl der Koordinatenpaare für Polyline, Polymarker, und Filled Area oder -1 (unbegrenzt)
work_out[15]: maximale Länge des intin-Array oder -1 (unbegrenzt)
work_out[16]: Anzahl der Maustasten
work_out[17]: Verfügbarkeit von Linientypen für breite Linien
0: nicht verfügbar
1: verfügbar
work_out[18]: Anzahl der Schreibmodi für breite Linien
work_out[19]: Clipping-Flag
0: Clipping aus
1: Clipping an
work_out[28]: Bezier-Flag. Bit 1 gibt Auskunft über die Bezierfähigkeiten
0: Keine Beziers
1: Beziers
work_out[45..48]: Clipping-Rechteck

Bemerkung:

Die Rückgabe des Clipping-Flags (work_out[19]) und des Clipping-Rechtecks (work_out[45..48]) ist GEM 2.x-kompatibel. Unter dem ATARI-VDI wird das Clipping-

Flag nicht zurückgegeben, wohl aber das Clipping-Rechteck (obwohl das nicht dokumentiert ist) - Benutzung also auf eigene Gefahr!
Die Rückgabe des Bezier-Flag (work_out[28], Bit 1) ist GEM/3-kompatibel. Im ATARI-VDI ist work_out[28] reserviert und enthält eine Null.

INQUIRE COLOR REPRESENTATION (VDI 26)

„**INQUIRE COLOR REPRESENTATION**“ gibt Auskunft über die eingestellten Farbintensitäten, wobei die Möglichkeit besteht, zwischen der übergebenen und der tatsächlich eingestellten Intensität zu unterscheiden.

Deklaration: int vq_color(int handle, int color_index,int set_flag, int *rgb);
Aufruf: valid = vq_color(handle, color_index, set_flag, rgb);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	26	vq_color
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	2	Einträge in intin
contrl[4]	4	Einträge in intout
contrl[6]	handle	
intin[0]	color_index	Farbnummer
intin[1] s	et_flag	Flag für Art der Intensität
intout[0]	valid	Farbindex außerhalb der Grenzen
intout[1..3]	rgb[0..2]	Intensität von Rot, Grün und Blau

Bedeutung von set_flag:

0: Es wird die vom Anwender an vs_color übergebene Farbintensität zurückgegeben.
1: Es wird die eingestellte Farbintensität zurückgegeben.

Bemerkung:

Die an vs_color übergebene und die eingestellte Farbintensität können voneinander abweichen, wenn die Anzahl der möglichen Farbabstufungen zu klein ist.

INQUIRE CURRENT POLYLINE ATTRIBUTES (VDI 35)

Diese Funktion gibt die aktuellen Linienattribute zurück.

Deklaration: void vql_attributes(int handle, int *attrib);

Aufruf: vql_attributes(handle, attrib);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	35	vql_attributes
contrl[1]	0	Einträge in ptsin
contrl[2]	1	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	5	Einträge in intout
contrl[6]	handle	
intout[0]	attrib[0]	Linientyp
intout[1]	attrib[1]	Linienfarbe
intout[2]	attrib[2]	Schreibmodus
intout[3]	attrib[4]	Linienanfangsform
intout[4]	attrib[5]	Linienendform
ptsout[0]	attrib[3]	Linienbreite

Bemerkung:

Beim ATARI-VDI werden intout[3..4] entgegen der VDI-Dokumentation nicht belegt. Die meisten Bindings gehen davon aus und liefern die zusätzlichen Parameter nicht mit.

INQUIRE CURRENT POLYMARKER ATTRIBUTES (VDI 36)

„**INQUIRE CURRENT POLYMARKER ATTRIBUTES**" gibt Auskunft über die eingestellten Markerattribute.

Deklaration: void vqm_attributes(int handle, int *attrib);
Aufruf: vqm_attributes(handle, attrib);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	36	vqm_attributes
contrl[1]	0	Einträge in ptsin
contrl[2]	1	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	3	Einträge in intout
contrl[6]	handle	
intout[0]	attrib[0]	Markertyp

intout[1]	attrib[1]	Markerfarbe
intout[2]	attrib[2]	Schreibmodus
ptsout[0]	attrib[4]	Markerbreite
ptsout[1]	attrib[3]	Markerhöhe

Bemerkung:

Die Markerbreite wird vom ATARI-VDI nicht zurückgegeben. Die meisten Bindings übergeben daher attrib[4] nicht.

INQUIRE CURRENT FILL AREA ATTRIBUTES (VDI 37)

Diese Funktion gibt Auskunft über die aktuellen Füllattribute.

Deklaration: void vqf_attributes(int handle, int *attrib);
Aufruf: vqf_attributes(handle, attrib);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	37	vqm_attributes
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	5	Einträge in intout
contrl[6]	handle	
intout[0]	attrib[0]	Fülltyp
intout[1]	attrib[1]	Füllfarbe
intout[2]	attrib[2]	Füllmusterindex
intout[3]	attrib[3]	Schreibmodus
intout[4]	attrib[4]	Umrahmungs-Flag

INQUIRE CURRENT GRAPHIC TEXT ATTRIBUTES (VDI 38)

Die gesetzten Textattribute werden von dieser Funktion geliefert.

Deklaration: void vqt_attributes(int handle, int *attrib);
Aufruf: vqt_attributes(handle, attrib);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	38	vqt_attributes
contrl[1]	0	Einträge in ptsin
contrl[2]	2	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	6	Einträge in intout
contrl[6]	handle	
intout[0]	attrib[0]	Zeichensatznummer
intout[1]	attrib[1]	Textfarbe
intout[2]	attrib[2]	Textrotation in 1/10 Grad
intout[3]	attrib[3]	horizontale Ausrichtung
intout[4]	attrib[4]	vertikale Ausrichtung
intout[5]	attrib[5]	Schreibmodus
ptsout[0]	attrib[6]	Zeichenbreite
ptsout[1]	attrib[7]	Zeichenhöhe
ptsout[2]	attrib[8]	Zeichenzellenbreite
ptsout[3]	attrib[9]	Zeichenzellenhöhe

Bemerkung:

Das ATARI-VDI gibt fehlerhafterweise in intout[5] den Schreibmodus -1 zurück. Im NVDI geschieht das nur bei eingeschalteter Fehlerkompatibilität.

INQUIRE TEXT EXTENT (VDI 116)

Mit „**INQUIRE TEXT EXTENT**" werden die minimalen Ausmaße eines Rechtecks berechnet, das die übergebene Zeichenkette umrahmt.
Die Koordinaten der vier Eckpunkte werden relativ zu einem Koordinatensystem ausgegeben, wobei die Punkte gegen den Uhrzeigersinn durchnummeriert sind. Der erste Punkt liegt ohne Textdrehung in der linken unteren Ecke des Textrechtecks.

Deklaration: void vqt_extent(int handle, char *string, int *extent);
Aufruf: vqt_extent(handle, string, extent);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	116	vqt_extent
contrl[1]	0	Einträge in ptsin
contrl[2]	4	Einträge in ptsout

contrl[3]	n	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]		handle
intin[0..n-1]	string[0..n-1]	Zeichenkette
ptsout[0..7]	extent[0..7]	Koordinaten des Textrechtecks

INQUIRE CHARACTER CELL WIDTH (VDI 117)

Die horizontalen Textausmaße werden von dieser Funktion berechnet.

Deklaration: int vqt_width(int handle, char character,int *cell_width, int *left_delta, int *right_delta);

Aufruf: status = vqt_width(handle, character, &cell_width,&left_delta, &right_delta);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	117	vqt_width
contrl[1]	0	Einträge in ptsin
contrl[2]	3	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[6]	handle	
intin[0]	character	Zeichennummer
intout[0]	status	Zeichennummer oder -1 (Fehler)
ptsout[0]	cell_width	Zeichenzellenbreite
ptsout[2]	left_delta	linker Abstand zur Zeichenzelle
ptsout[4]	right_delta	rechter Abstand zur Zeichenzelle

Bemerkung:

Um die Breite einer Zeichenkette zu ermitteln, ist der Aufruf von vqt_extent zu empfehlen - das Addieren der von vqt_width zurückgelieferten Breiten ergibt nicht immer die Gesamtbreite einer Zeichenkette!

INQUIRE FACE NAME AND INDEX (VDI 130)

„**INQUIRE FACE NAME AND INDEX**“ gibt Auskunft über Index und Namen eines Zeichensatzes.

Deklaration: int vqt_name(int handle, int element_num, char *name);
Aufruf: index = vqt_name(handle, element_num, name);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	130	vqt_name
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	33	Einträge in intout
contrl[6]	handle	
intin[0]	element_num	Nummer (1 bis Maximalanzahl)
intout[0]	index	Zeichensatznummer
intout[1..16]	name[0..15]	Zeichensatzname
intout[17..32]	name[16..32]	Zeichensatzattribute

INQUIRE INPUT MODE (VDI 115)

Diese Funktion ermittelt den Eingabemodus eines Gerätes.

Deklaration: void vqin_mode(int handle, int dev_type, int *input_mode);
Aufruf: vqin_mode(handle, dev_type, &input_mode);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	115	vqin_mode
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[6]	handle	
intin[0]	dev_type	Gerätenummer (siehe vsin_mode)
intout[1]	input_mode	Eingabe-Modus

INQUIRE CURRENT FACE INFORMATION (VDI 131)

Es wird Auskunft über den aktuellen Zeichensatz gegeben, wobei Attribute und Vergrößerung/Verkleinerung berücksichtigt werden.

Deklaration: void vqt_fontinfo(int handle, int *minADE, int *maxADE,int *distances, int *maxwidth, int *effects);

Aufruf: vqt_fontinfo(handle, &minADE, &maxADE, distances,&max_width, effects);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	131	vqt_fontinfo
contrl[1]	0	Einträge in ptsin
contrl[2]	5	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	2	Einträge in intout
contrl[6]	handle	
intout[0]	minADE	niedrigste Zeichennummer
intout[1]	maxADE	höchste Zeichennummer
ptsout[0]	maxwidth	Maximale Zeichenzellenbreite
ptsout[1]	distances[0]	
ptsout[2]	effects[0]	
ptsout[3]	distances[1]	
ptsout[4]	effects[1]	
ptsout[5]	distances[2]	
ptsout[6]	effects[2]	
ptsout[7]	distances[3]	
ptsout[9]	distances[4]	

Bedeutung von distances:

distances[0]: Abstand von der Untergrenze der Zeichenzelle zur Basislinie
distances[1]: Abstand der Unterlänge zur Basislinie
distances[2]: Abstand der Halblinie zur Basislinie
distances[3]: Abstand der Zeichenobergrenze zur Basislinie
distances[4]: Abstand der Zeichenzellenobergrenze zur Basislinie

Bedeutung von effects:

effects[0]: Verbreiterung bei Texteffekten
effects[1]: linker Abstand bei Kursivschrift
effects[2]: rechter Abstand bei Kursivschrift

INQUIRE JUSTIFIED GRAPHHICS TEXT (VDI 132)

Diese Funktion gibt für jedes übergebene Zeichen die X- und Y-Offsets von einem gegebenen Ursprung aus zurück.

Deklaration: void vqt_justified(int handle, int x, int y, char *string,int length, int word_space, int char_space, int *offsets);

Aufruf: vqt_justified(handle, x, y, string, length, word_space,char_space, offsets);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	132	vqt_justified
contrl[1]	2	Einträge in ptsin
contrl[2]	n	Einträge in ptsout
contrl[3]	2+n	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[6]	handle	
intin[0]	word_space	Flag für Wortzwischenraumdehnung
intin[1]	char_space	Flag für Zeichenzwischenraumdehnung
intin[2..n+1]	string[0..n-1]	Zeichenkette
ptsin[0]	x	
ptsin[1]	y	
ptsin[2]	length	horizontale Textlänge
ptsout[0..2n-1]	offsets[0..2n-1]	Abstands-Array

Bemerkung:

Dies ist eine GEM 2.x-Funktion, die unter dem ATARI-VDI nicht verfügbar ist.

7. Escape-Funktionen (VDI 5)

Die ESCAPE-Funktionen ermöglichen es, spezielle Fähigkeiten des Gerätes auszunutzen.

INQUIRE ADDRESSABLE ALPHA CHARACTER CELLS (VDI 5, ESCAPE 1)

Diese Funktion gibt über die Anzahl der Zeilen und Spalten des Textbildschirmes Auskunft.

Deklaration: void vq_chcells(int handle, int *rows, int *columns);
Aufruf: vq_chcells(handle, &rows, &columns);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	5	ESCAPE
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	2	Einträge in intout
contrl[5]	1	vq_chcells
contrl[6]	handle	
intout[0]	rows	Zeilenanzahl
intout[1]	columns	Spaltenanzahl

EXIT ALPHA MODE (VDI 5, ESCAPE 2)

„**EXIT ALPHA MODE**“ schaltet den Textmodus aus.

Deklaration: void v_exit_cur(int handle);
Aufruf: v_exit_cur(handle);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	5	ESCAPE
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	2	v_exit_cur
contrl[6]	handle	

ENTER ALPHA MODE (VDI 5, ESCAPE 3)

Mit dieser Funktion gelangt man in den Textmodus. Der Bildschirm wird gelöscht und der Text-Cursor in die linke obere Ecke gesetzt.

Deklaration: void v_enter_cur(int handle);
Aufruf: v_enter_cur(handle);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	5	ESCAPE
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	3	v_enter_cur
contrl[6]	handle	

ALPHA CURSOR UP (VDI 5, ESCAPE 4)

Der Text-Cursor wird von dieser Funktion eine Zeile nach oben bewegt. Sofern er sich in der obersten Zeile befindet, geschieht nichts.

Deklaration: void v_curup(int handle);
Aufruf: v_curup(handle);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	5	ESCAPE
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	4	v_curup
contrl[6]	handle	

ALPHA CURSOR DOWN (VDI 5, ESCAPE 5)

Der Text-Cursor wird um eine Zeile nach unten bewegt. Befindet sich der Cursor in der untersten Zeile, so passiert nichts.

Deklaration: void v_curdown(int handle);
Aufruf: v_curdown(handle);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	5	ESCAPE
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	5	v_curdown
contrl[6]	handle	

ALPHA CURSOR RIGHT (VDI 5, ESCAPE 6)

„ALPHA CURSOR RIGHT" bewegt den Text-Cursor nach rechts, wobei am Zeilenende nichts geschieht.

Deklaration: void v_curright(int handle);
Aufruf: v_curright(handle);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	5	ESCAPE
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	6	v_curright
contrl[6]	handle	

ALPHA CURSOR LEFT (VDI 5, ESCAPE 7)

Der Text-Cursor wird von dieser Funktion nach links bewegt. Es passiert nichts, wenn er sich bereits am Zeilenanfang befinden sollte.

Deklaration: void v_curleft(int handle);
Aufruf: v_curleft(handle);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	5	ESCAPE
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	7	v_curleft
contrl[6]	handle	

HOME ALPHA CURSOR (VDI 5, ESCAPE 8)

Von dieser Funktion wird der Text-Cursor in die linke obere Ecke gesetzt.

Deklaration: void v_curhome(int handle);
Aufruf: v_curhome(handle);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	5	ESCAPE
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	8	v_curhome
contrl[6]	handle	

ERASE TO END OF ALPHA SRCEEN (VDI 5, ESCAPE 9)

Der Bildschirm wird von der aktuellen Cursor-Position ab gelöscht.

Deklaration: void v_eeos(int handle);
Aufruf: v_eeos(handle);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	5	ESCAPE
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	9	v_eeos
contrl[6]	handle	

ERASE TO END OF ALPHA TEXT LINE (VDI 5, ESCAPE 10)

Diese Funktion löscht ab der Cursor-Position die aktuelle Zeile.

Deklaration: void v_eeol(int handle);
Aufruf: v_eeol(handle);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	5	ESCAPE
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	10	v_eeol
contrl[6]	handle	

DIRECT ALPHA CURSOR ADDRESS (VDI 5, ESCAPE 11)

Mit „DIRECT ALPHA CURSOR ADDRESS“ kann der Text-Cursor direkt positioniert werden.

Deklaration: void v_curaddress(int handle, int row, int column);
Aufruf: v_curadress(handle, row, column);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	5	ESCAPE
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	2	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	11	v_curaddress
contrl[6]	handle	
intin[0]	row	Zeile
intin[1]	column	Spalte

OUTPUT CURSOR ADDRESSABLE ALPHA TEXT (VDI 5, ESCAPE 12)

Diese Funktion gibt an der aktuellen Cursor-Position eine Zeichenkette aus.

Deklaration: void v_curtext(int handle, char *string);
Aufruf: v_curtext(handle, string);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	5	ESCAPE
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	n	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	12	v_curtext
contrl[6]	handle	
intin[0..n-1]	string[0..n-1]	Zeichenkette

REVERSE VIDEO ON (VDI 5, ESCAPE 13)

Diese Funktion schaltet auf inverse Textausgabe um.

Deklaration: void v_rvon(int handle);
Aufruf: v_rvon(handle);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	5	ESCAPE
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	13	v_rvon
contrl[6]	handle	

REVERSE VIDEO OFF (VDI 5, ESCAPE 14)

Die inverse Textausgabe wird ausgeschaltet.

Deklaration: void v_rvoff(int handle);
Aufruf: v_rvoff(handle);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	5	ESCAPE
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	14	v_rvoff
contrl[6]	handle	

INQUIRE CURRENT ALPHA CURCOR ADDRESS (VDI 5, ESCAPE 15)

Von dieser Funktion wird die Position des Text-Cursors zurückgegeben.

Deklaration: void vq_curaddress(int handle, int *row, int *column);
Aufruf: vq_curaddress(handle, &row, &column);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	5	ESCAPE
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	2	Einträge in intout
contrl[5]	15	vq_curaddress
contrl[6]	handle	
intout[0]	row	Zeile
intout[1]	column	Spalte

INQUIRE TABLET STATUS (VDI 5, ESCAPE 16)

„**INQUIRE TABLET STATUS**" gibt Auskunft über die Verfügbarkeit einer Maus, eines Joysticks oder eines Grafik-Tablettes.

Deklaration: int vq_tabstatus(int handle);
Aufruf: status = vq_tabstatus(handle);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	5	ESCAPE
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[5]	16	vq_tabstatus
contrl[6]	handle	
intout[0]	status	0: nicht verfügbar, 1: verfügbar

HARD COPY (VDI 5, ESCAPE 17)

Der Bildschirminhalt wird auf dem Drucker ausgegeben (XBIOS 36).

Deklaration: void v_hardcopy(int handle);
Aufruf: v_hardcopy(handle);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	5	ESCAPE
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	17	v_hardcopy
contrl[6]	handle	

PLACE GRAPHIC CURSOR AT LOCATION (VDI 5, ESCAPE 18)

Mit dieser Funktion kann man den Mauszeiger einschalten und positionieren.

Deklaration: void v_dspcur(int handle, int x, int y);
Aufruf: v_dspcur(handle, x, y);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	5	ESCAPE
contrl[1]	1	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	18	v_dspcur
contrl[6]	handle	
ptsin[0]	x	
ptsin[1]	y	

Bemerkung:

Im ATARI-VDI wird der Mauszeiger eingeschaltet, nicht jedoch positioniert.

REMOVE LAST GRAPHIC CURSOR (VDI 5, ESCAPE 19)

Diese Funktion entfernt den Mauszeiger.

Deklaration: void v_rmcur(int handle);
Aufruf: v_rmcur(handle);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	5	ESCAPE
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	0	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	19	v_rmcur
contrl[6]	handle	

GENERATE SPECIFIED TONE (VDI 5, ESCAPE 61)

Von dieser Funktion wird ein Ton in angegebener Höhe und Länge erzeugt.

Deklaration: void v_sound(int handle, int frequency, int duration);
Aufruf: v_sound(handle, frequency, duration);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	5	ESCAPE
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	2	Einträge in intin
contrl[4]	0	Einträge in intout
contrl[5]	61	v_sound
contrl[6]	handle	
intin[0]	frequency	Frequenz in Hertz
intin[1]	duration	Dauer in Timer-Ticks (max. 255)

Bemerkung:

Dies ist eine GEM 2.x-Funktion, die unter dem ATARI-VDI nicht verfügbar ist.

SET/CLEAR MUTING FLAG (VDI 5, ESCAPE 62)

Das Tonflag wird gelöscht, gesetzt oder der Status wird zurückgeliefert.

Deklaration: int vs_mute(int handle, int action);
Aufruf: status = vs_mute(handle, action);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	5	ESCAPE
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	1	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[5]	62	vs_mute
contrl[6]	handle	
intin[0]	action	
intout[0]	status	Tonerzeugungs-Status

Bedeutung von action:

-1: Status zurückgeben
0: Tonerzeugung aus
1: Tonerzeugung an

Bemerkung:

Dies ist eine GEM 2.x-Funktion, die unter dem ATARI-VDI nicht verfügbar ist.

SET BEZIER QUALITY (VDI 5, ESCAPE 99)

Mit dieser Funktion wird die Qualität (und somit auch die Geschwindigkeit) der Bezierfunktionen beeinflußt.

Deklaration: int v_bez_qual(int handle, int prcnt, int *actual);
Aufruf: v_bez_qual(handle, prcnt, &actual);

Parameter:

Variable	Belegung	Bedeutung
contrl[0]	5	ESCAPE
contrl[1]	0	Einträge in ptsin
contrl[2]	0	Einträge in ptsout
contrl[3]	3	Einträge in intin
contrl[4]	1	Einträge in intout
contrl[5]	99	
contrl[6]	handle	
intin[0]	32 intin[0..1]	signalisieren v_bez_qual()
intin[1]	1	
intin[2]	prcnt	gewünschte Bezierqualität in Prozent
intout[0]	actual	eingestellte Bezierqualität in Prozent

Bemerkung:

Das Binding liefert in Register d0 ebenfalls den in <acutal> abgelegten Wert zurück.

Die Programmierschnittstelle des NVDI

Die Programmierschnittstelle des NVDI wurde über einen Cookie (deutsch: „Keks“) realisiert. Man findet den Zeiger auf den Cookie Jar (deutsch: „Keksdose“) in der dokumentierten Variablen $5a0. Dies ist eine Systemvariable, auf die Sie nur im Supervisiormodus zugreifen können. Falls kein Cookie Jar existiert, enthält die Variable Null.

Der Cookie Jar enthält Langwortpaare, bestehend aus einer speziellen Kennung (dem „Cookie“ mit 4 Bytes Länge) und einem nachfolgenden Langwort (4 Bytes).

```
typedef struct
{
        char cookie_id[4];      /* enthält im NVDI 0x4e564449 = 'NVDI'      */
        long cookie_value;      /* zeigt bei NVDI auf die NVDI-Struktur     */
}COOKIE;
```

Der cookie_value zeigt beim NVDI auf die folgende Struktur:

```
typedef struct
{
        unsigned int  nvdi_version;     /* z.B. $0001 für Version 0.01      */
        unsigned long nvdi_datum;       /* z.B. $18061990 für 18.06.1990    */
struct
{
        unsigned            : 9;        /* reserviert                       */
        unsigned alert      : 1;        /* Fehlermeldungen ein/aus          */
        unsigned            : 1;        /* reserviert                       */
        unsigned linea      : 1;        /* LINE-A                           */
        unsigned mouse      : 1;        /* Dynamische Maus                  */
        unsigned gemdos     : 1;        /* GEMDOS-Zeichenausgabe            */
        unsigned fehler     : 1;        /* Fehlerkompatibilität             */
        unsigned gdos       : 1;        /* GDOS                             */
} nvdi_config;
}NVDI_STRUC;
```

Bedeutung der Bits von nvdi_config (Bitnummer):

0: GDOS
1: Fehlerkompatibilität
2: (schnelle) GEMDOS-Zeichenausgabe
3: Dynamische Maus
4: LINE-A
6: Fehlermeldungen

Durch Setzen der entsprechenden Bits im Konfigurationswort können verschiedene Schalter (wie im NVDICONF.ACC) aktiviert oder (durch Löschen) deaktiviert werden. Die reservierten Bits dürfen nicht verändert werden!

Wie findet man einen Cookie?

```
/* Routine zum Auffinden eines Cookie                                          */
/* Rückgabewert: FALSE, wenn kein Cookie Jar installiert oder Cookie nicht gefunden. */
/* TRUE, wenn Cookie gefunden. Eintrag des cookie_value    in &val.            */

#include <TOS.H>

/* Typdefinitionen                                                             */
typedef struct
{
        char cookie_id[4];          /* enthält bei NVDI 0x4e564449 = 'NVDI'    */
        long cookie_value;          /* zeigt bei NVDI auf die NVDI-Struktur    */
} COOKIE;
typedef enum{FALSE,TRUE};

/* Funktionsprototypen                                                         */
int  getcookie(long id, long *val);
long get_jar(void);
int  getcookie(long id, long *val)
{
        COOKIE *cookie_jar;
        cookie_jar = (COOKIE *)Supexec( get_jar );      /* Cookie-Jar auslesen     */
        if( cookie_jar )                                /* Cookie-Jar ist vorhanden */
        {
          do
          {
                if( *((long *) &cookie_jar->cookie_id) == id )
                {
                        *val = cookie_jar->cookie_value;
                        return(TRUE);                   /* Cookie gefunden         */
                }
          }
          while( (*(long *) &(cookie_jar++)->cookie_id) );  /* NULL = Ende         */
        }

        return(FALSE);                                  /* Cookie nicht gefunden   */
}
long get_jar(void)                              /* hole Zeiger auf den Cookie-Jar  */
{
        return( (long)(*((COOKIE **)0x5a0L)) );
}
```

Weitere Anwendungen finden Sie auf der Orginaldiskette.

VDI-Programmierung in Assembler

Der Aufruf des VDI erfolgt über den TRAP #2. In Register d0 muß sich der Opcode #115 befinden. Register d1 zeigt auf folgende Struktur:

```
typedef struct
{
        int *control;
        int *intin;
        int *ptsin;
        int *intout;
        int *ptsout;
}PB
```

Im VDI werden die Register d1-d7/a0-a6 nicht verändert, d0 wird zerstört. Bei GEMDOS-, BIOS- und XBIOS-Aufrufen werden die Inhalte der Register d0-d2/a0-a2 zerstört, was der der üblichen ATARI-Konvention entspricht.

Die XBRA-Kennung von NVDI lautet 'NVDI'.

Programmiertips

Zuerst soll hier auf einige, immer wieder verwechselte oder falsch gedeutete Begriffe eingegangen werden, deren Verständnis zur effektiven VDI-Programmierung nicht ganz unwichtig ist.

Welcher Unterschied besteht zwischen einer physikalischen und einer virtuellen Workstation?

Mit dem Begriff physikalische Workstation bezeichnet man existierende Ein- und Ausgabegeräte, die i.a. nicht von mehreren Programmen gleichzeitig benutzt werden können (z.B. Drucker).

Virtuelle Workstations dagegen sind auf einer physikalischen Workstation mehrfach vorhanden und ermöglichen so z.B. die Koexistenz von Programmen und Accessories bei der Bildschirmausgabe (jedes Programm bzw. Accessory hat eine (oder auch mehrere) eigene virtuelle Workstation(s), deren Parameter - z.B. Textgröße, Linienfarbe,... - von anderen Programmen/Accessories nicht beeinflußbar sind).

Was ist eigentlich der Unterschied zwischen einer Workstation-ID und einem Workstation-Handle?

Hinter der Workstation-ID verbirgt sich die Geräteidentifikationsnummer eines Treibers. Diese ID muß zum Öffnen einer physikalischen Workstation (z.B. Drucker, Plotter,...) in work_in[0] übergeben werden. Beispiel für Workstation-IDs siehe Abschnitt 'Die ASSIGN.SYS-Datei'.

Das Workstation-Handle hingegen ist eine Kennung, die nach dem Öffnen einer Workstation vom VDI zurückgegeben wird und es ermöglicht, die Ausgaben mehrerer Applikationen (z.B. Accessories und Hauptprogramm) zu koordinieren.

Was ist eigentlich mit LINE-A?

Seitdem die LINE-A-Funktionen von ATARI dokumentiert wurden, ranken sich die abenteuerlichsten Gerüchte um die Geschwindigkeit und die Möglichkeiten dieser Low-Level-Grafikschnittstelle.

Tatsache ist:

- Die Benutzung von LINE-A-Funktionen ist NICHT schneller als die Benutzung von VDI-Funktionen. Ganz im Gegenteil: Im NVDI sind LINE-A-Funktionen wesentlich langsamer als äquivalente VDI-Funktionen.
- LINE-A-Aufrufe werden auf den meisten Grafikkarten mit Grafikprozessor oder anderer Bildschirmorganisation NICHT unterstützt.
- Die Benutzung von mehr als 16 Farben über LINE-A ist NICHT dokumentiert. Somit kann z.B. in der niedrigen TT-Auflösung nicht mit LINE-A gearbeitet werden.
- Die Benutzung von LINE-A ist NICHT einfacher als die Benutzung des VDI. Als abschreckendes Beispiel sei die LINE-A-Funktion TextBlt genannt - sie ist nicht nur wesentlich langsamer als die VDI-Funktion v_gtext, sondern auch sehr umständlich anzusprechen.
- Veränderungen in den sogenannten LINE-A-Variablen müssen NICHT das Verhalten des VDI-Bildschirmtreibers ändern.

Die negativen LINE-A-Variablen dürfen grundsätzlich nur ausgelesen und NICHT verändert werden.
Ausnahmen: Bildschirmtreiber oder Grafikerweiterungen (z.B. AutoSwitch OverScan)
- Mit LINE-A ist paralleler Betrieb von mehreren Monitoren NICHT möglich.
- LINE-A ermöglicht NICHT auflösungsunabhängiges Programmieren.

Fazit:
Aufrufe von LINE-A-Funktionen haben in sauberen Programmen nichts zu suchen. Fast alles, was über LINE-A-Aufrufe machbar ist, kann auch über das VDI erfolgen (oft sogar wesentlich einfacher).

Was sollte man mit dem VDI nicht anstellen?

1. Vermischen Sie nie VDI- mit Line-A-Aufrufen.
2. Erwarten Sie nicht, daß Aufrufe von VDI-Funktionen Line-A-Variablen in bestimmter Weise beeinflussen.

Verwendung von VDI-Funktionen in AUTO-Ordner-Programmen

Auf die Frage 'Kann man GEM-Funktionen in AUTO-Ordner-Programmen aufrufen?' bekommt man im allgemeinen die Standardantwort 'Nein, unmöglich!'. Wie bei Standardantworten so üblich, ist das aber nur die halbe Wahrheit. Richtig ist, daß man in AUTO-Ordner-Programmen keine AES-Funktionen aufrufen kann, da das AES erst nach Abarbeitung des AUTO-Ordners gestartet wird. Man kann jedoch die VDI-Funktionen aufrufen! Dazu muß man folgendermaßen vorgehen:

1. Feststellen, ob das Programm im AUTO-Ordner läuft. Dazu sollte man folgende Funktion aufrufen:

```
;
; int vq_aes(void);
; liefert:   -1, wenn AES installiert ist
;             0, wenn AES nicht installiert ist
; -> Programm läuft im AUTO-Ordner
vq_aes:    lea        vq_aes_pb(pc),a0
           move.l     a0,d1
           clr.w      aes_global        ;global[0] löschen
           move.w     #200,d0
           trap       #2
           move.w     aes_global(pc),d0  ;AES vorhanden?
           sne.b      d0
           ext.w      d0
rts

vq_aes_pb:  DC.L init_contrl
DC.L aes_global
DC.L aes_intin
DC.L aes_intout
DC.L aes_addrin
```

```
DC.L        aes_addrout
init_contrl: DC.W 10,0,1,0,0

aes_global:DS.W 15   ;global
aes_intin:  DS.W 2   ;intin
aes_intout: DS.W 1   ;intout
aes_addrin:   DS.L 1 ;addrin
aes_addrout: DS.L 1  ;addrout
```

Sollte das AES schon installiert sein, so darf die physikalische Workstation nicht geöffnet werden (-> Programmabbruch oder virtuelle Workstation öffnen,...).

2. Öffnen einer physikalischen Workstation (mit VDI 1). Das zurückgegebene Handle merken.

3. Wer mehr als eine Workstation benötigt (in AUTO-Ordner-Programmen wohl eher selten), der sollte mit dem unter 2. ermittelten Handle eine (oder mehrere) virtuelle Workstation(s) (mit VDI 100) öffnen.

4. Bei der Beendigung Ihres Programmes müssen, wenn vorher geöffnet, zuerst die virtuellen Workstations (mit VDI 101) geschlossen werden.

5. Der letzte Schritt ist das Schließen der physikalischen Workstation (mit VDI 2).

Punkt 5 darf auf keinen Fall vergessen werden, da sonst das AES nicht gestartet werden kann! Als Beispielanwendung mag das Programm A_PAINT auf der Orginaldiskette dienen.

„Schnelle“ (direkte) Bildschirmausgabe statt VDI-Ausgabe?

Manche Programme bieten eine Option im Monochrommodus direkt in den Bildschirmspeicher zu schreiben, um gegenüber der VDI-Ausgabe einen Geschwindigkeitsvorteil zu erzielen. Die Praxis hat gezeigt, daß diese Direktausgabe meist nur 10-20% schneller als die Ausgabe über NVDI ist (wobei sich der Unterschied durch geschicktere VDI-Programmierung noch deutlich verringern ließe).

Bei intelligenter Grafik-Hardware (Grafikprozessor) kann die Direktausgabe beträchtlich langsamer als die Ausgabe über einen optimierten VDI-Treiber sein, da die direkte Hardware-Unterstützung durch ein Anwenderprogramm bei der Vielzahl der Grafikkarten sehr unwahrscheinlich ist.

Fazit:

Sparen Sie sich die Mühe eine Direktausgabe zu implementieren und investieren Sie die gewonnene Zeit in andere Programmfunktionen!

Wie erkennt erkennt man, ob ein GDOS Bezierunterstützung bietet?

Mit NVDI genügt es, die VDI-Funktion vq_extnd() aufzurufen und anschließend Bit 1 von work_out[28] zu überprüfen. Ist das Bit 1 gesetzt, bietet das GDOS Bezierunterstützung, andernfalls nicht.

Leider ist die Funktion vq_extnd() bei anderen GDOSsen mit Bezierunterstützung (noch) nicht in dieser Richtung erweitert worden, so daß hier nur ein kleiner Trick weiterhilft:
- Die Funktion v_bez_qual() mit einer Bezierqualität von 100% aufrufen.
- intout[0] auf Null setzen und die Funktion v_bez_on() aufrufen.
- Den Rückgabewert von v_bez_on() merken und anschließend v_bez_off() aufrufen.

Falls keine Bezierunterstützung vorhanden ist, wird der Rückgabewert von v_bez_on() Null sein, andernfalls ungleich Null.

Das VDI-Standardformat: „Wieso, wozu, warum?“

Das Standardformat und die über vr_trnfm() mögliche Umwandlung ins gerätespezifische Format ermöglichen die Ausgabe auf Grafikkarten mit verschiedenartiger Grafikorganisation. Möchte man beispielsweise einen Bildausschnitt in der TT-niedrig-Auflösung speichern, so sollte man ihn mit vro_cpyfm() ausschneiden, mit vr_trnfm() ins Standardformat transformieren und anschließend möglichst als XIMG abspeichern. Wenn dieses Bild nun z.B. auf einer VGA-Karte mit 256 Farben angezeigt werden soll, so muß man es mit vr_trnfm() in das gerätespezifische Format der VGA-Karte transformieren und dann mittels vro_cpyfm() ausgeben.

Wenn man den Zwischenschritt über vr_trnfm() ausläßt, kann man auf der VGA-Karte nur wundersamen Pixelmüll erkennen, da in TT-niedrig die Grafik in sogenannten „Interleaved Planes“ organsiert ist, während bei VGA-Karten ein „Packed-Pixel“-Format benutzt wird.

Bei der Benutzung von vr_trnfm() ist noch anzumerken, daß man, wenn der Speicher ausreicht, verschiedene Quell- und Zielbuffer benutzen sollte. Bei gleicher Quell- und Zielbufferadresse wird die Umwandlung für vr_trnfm() relativ aufwendig und beansprucht sehr viel Zeit.

VDI-Programmierung auf HiColor/TrueColor-Karten

Bei Grafikkarten mit mehr als 256 Farben ist keine „Color LookUp Table“ (CLUT) vorhanden. Da die VDI Farbeinstellfunktionen aber index-orientiert sind und der maximale Index auf 32767 beschränkt ist, ist bei HiColor- und TrueColor-Karten ein etwas anderes Vorgehen notwendig, um die Möglichkeiten der Karte auszunutzen.

- Das VDI stellt für jede virtuelle Workstation eine lokale Pseudo-Farbpalette (mit 256 Einträgen) zur Verfügung. Es stehen somit 256 „Farbstifte“ zur Verfügung, die über vs_color() beliebig verändert werden können. Da keine CLUT mehr vorhanden ist, beeinflussen Veränderungen in der Pseudo-Farbpalette die Farbe der bereits gezeichneten Pixel oder die Farbpaletten anderer virtueller Workstations nicht.

- Als Anzahl der Farben (v_opnvwk(), work_out[13]) wird 256 zurückgeliefert. Dies entspricht der Anzahl der „Farbstifte“ - ein Rückschluß auf die Anzahl der Planes ist NICHT zulässig.

- Die Anzahl der Planes (vq_extnd(), work_out[4]) dagegen berücksichtigt den tatsächlichen Speicherbedarf. Also:
  16 Planes bei 32k- oder 65k-Farben
  24 Planes bei True Color
  32 Planes bei True Color mit Overlay Byte

- Da keine CLUT mehr vorhanden ist, repräsentieren die Pixel im Speicher direkt einen RGB-Wert.
  Die Bit-Organisation kann je nach verwendeter Hardware beträchtlich differieren (bei VGA-Grafikarten bspw. liegen die Pixel im Intel-Format vor). Wenn Ihr Programm selbst Raster erzeugt, sind Sie nur bei Verwendung des VDI Standard-Formats und der vr_trnfm()-Funktion auf der „sicheren Seite".
  Alternativ ist es natürlich möglich mit Hilfe der Rasterfunktionen den tatsächlich vorhandenen Bit-Aufbau zu ermitteln und dann mit geeigneten Routinen die Raster bereits im geräteabhängigen Format zu erzeugen. Dies kann zu einem beträchtlichen Geschwindigkeitsgewinn führen, zieht aber einen gewissen Programmieraufwand nach sich und ist nicht ganz trivial.

- v_getpixel() liefert Pixelwerte zurück, die größer als ein Byte sind. Falls der Pixelwert nicht in der Pseudopalette vorhanden ist, wird als Farbindex -1 zurückgegeben. Bei True-Color-Karten wird v_get_pixel() grundsätzlich -1 zurückliefern, da die Pixelwerte nicht in einem Wort darstellbar sind.

VDI-Benutzung in Accessories

Die Benutzung von VDI-Funktionen ist in Accessories unter TOS kritisch, da sie keine eigenständigen Prozesse sind. Das führt dazu, daß der Speicher, der von einigen VDI-Funktionen angefordert werden muß, nicht dem Accessory, sondern der gerade aktiven Hauptapplikation gehört. Betroffen davon sind z.B. die VDI-Funktionen v_opnwk(), v_opnvwk(), vst_load_fonts().

Wird die Hauptapplikation beendet, so wird sämtlicher Speicher der Hauptapplikation an das Betriebssystem zurückgegeben. Der Speicher für die Verwaltungsinformationen über die Workstation des Accessories ist jetzt freigegeben und kann bei der nächsten Speicheranforderung des jetzt aktiven Hauptprogramms manipuliert werden. Da das VDI von dieser Speicherfreigabe nichts mitbekommt, kann es bei einer der nächsten VDI-Aufrufe im Accessory aufgrund beschädigter Daten zu einem Systemabsturz kommen.

Es gibt vier mögliche Vorgehensweisen, um dieses Problem in den Griff zu bekommen:

1. TOS schickt vor dem Beenden der Hauptapplikation eine AES-Nachricht (AC_CLOSE) an die Accessories. Das Accessory muß jetzt so schnell wie möglich seine VDI-Workstation schließen.
   Leider wird die AC_CLOSE-Nachricht erst in TOS 2.xx, TOS 3.xx Versionen rechtzeitig an die Accessories verschickt. In älteren TOS-Versionen trifft die Nachricht zu spät (nach Beenden der Hauptapplikation) ein, so daß hier wieder die Gefahr eines Systemabsturzes droht.

2. NVDI läßt sich über das Programm NVDIDFLT so konfigurieren, daß NVDI beim Anfordern von VDI-Speicher den Prozeßdeskriptor verbiegt. Dadurch gehört dieser Speicher NVDI und nicht dem aktuellen Prozeß. Da andere GDOSse diese Option nicht bieten und das Verbiegen des Prozeßdeskriptors von ATARI nicht dokumentiert ist, ist diese Methode nur eingeschränkt zu empfehlen - auf KEINEN Fall sollte das Verbiegen des Prozeßdeskriptors vom Accessory selbst durchgeführt werden!

3. Das Accessory sperrt sofort nachdem es geladen wurde mit einer AES-Funktion den Bildschirm (dadurch wird unter TOS das Taskswitching unterbrochen), öffnet die virtuelle Workstation und lädt ggf. noch Zeichensätze, schließt jedoch niemals die VDI-Workstation.

Problematisch bei dieser Methode ist der Auflösungswechsel unter neueren TOS-Versionen (TOS 2.xx/3.xx). Einige GDOSse erwarten, daß beim Schließen der physikalischen Bildschirm-Workstation durch das AES alle virtuellen Workstations geschlossen und der Zeichensatzspeicher durch vst_unload_fonts() freigegeben wurde. Geschieht dies nicht (weil z.B. ein Accessory nie vst_unload_fonts() und v_clsvwk() aufgerufen hat), kann es nach dem Auflösungswechsel beim ersten Zugriff auf GDOS-Zeichensätze zum Systemabsturz kommen.

NVDI vermeidet dieses Problem beim Auflösungswechsel,indem es beim Schließen der physikalischen Bildschirm-Workstation selbständig alle noch nicht geschlossenen virtuellen Workstations schließt und ggf. den Zeichensatzspeicher freigibt.

4. Sie verwenden das **Multitasking-System Mag!X**. Hier und (voraussichtlich) auch in dem in derzeit noch in der Entwicklung befindlichen MultiTOS von ATARI sind Accessories eigenständige Prozesse, die selbst Speicher anfordern können, der ihnen dann auch gehört.

Das Zeichensatzformat

Die Zeichenausgabe des ATARI ST erfolgt bekanntlich ausschließlich im Grafikmodus, wodurch die Verwendung verschiedener Schriftarten und Textgrößen erst möglich wurde. Ein besonderes Problem stellt dabei die Verwendung von Proportionalschriften dar, bei denen die Zeichenbreiten ständig variieren. Gelöst wurde das Problem von den GEM-Designern dadurch, daß alle Zeichen nebeneinander ausgegeben wurden. Durch diese Aneinanderreihung ergibt sich ein „Zeichensatzbild", dessen Höhe der Zeichenhöhe und dessen Breite der Summe aller Zeichenbreiten entspricht. Diese Art der Anordnung hat den Vorteil eines geringstmöglichen Speicherbedarfs, aber den Nachteil, daß die Position oft (aber nicht immer - monospaced Fonts) mit Hilfe einer speziellen Tabelle (Char-Offset-Tabelle) berechnet werden muß und innerhalb der Textausgaberoutine viele Prozessoroperationen notwendig sind, bis die Zeichenausgabe erfolgen kann.

Ein GEM-Zeichensatz beginnt mit dem folgenden Header:

```
typedef struct font_hdr
{
        int            font_id;             /* Fontnummer                      */
        int            point;               /* Fontgröße in Punkten            */
        char           name[32];            /* Name                            */
        unsigned int   first_ade;           /* kleinster ASCII-Wert            */
        unsigned int   last_ade;            /* größter ASCII-Wert              */
        unsigned int   top;                 /* Abstand Top <-> Baseline        */
        unsigned int   ascent;              /* Abstand Ascent <-> Baseline     */
        unsigned int   half;                /* Abstand Half <-> Baseline       */
        unsigned int   descent;             /* Abstand Descent <-> Baseline    */
        unsigned int   bottom;              /* Abstand Bottom <-> Baseline     */
        unsigned int   max_char_width;      /* maximale Zeichenbreite          */
        unsigned int   max_cell_width;      /* maximale Zeichenzellenbreite    */
        unsigned int   left_offset;         /* linker Offset bei Kursivschrift */
        unsigned int   right_offset;        /* rechter Offset bei Kursivschrift */
        unsigned int   thicken;             /* Verbreiterung bei Fettschrift   */
        unsigned int   ul_size;             /* Dicke der Unterstreichung       */
        unsigned int   lighten;             /* Maske für Hellschrift           */
        unsigned int   skew;                /* Maske für Kursivschrift         */
        struct
        {
            unsigned               :12;     /* reserviert                      */
            unsigned   mono_spaced : 1;     /* 0: Proportional/ 1: Äquidistant */
            unsigned   f68000      : 1;     /* 0: Intel-/ 1: Motorola-Format   */
            unsigned   hor_of_tab  : 1;     /* 1: HOT verwenden                */
            unsigned   system      : 1;     /* 1: Systemfont                   */
        } flags;
        unsigned char  *hor_table;          /* Horizontal Offset Table         */
        unsigned int   *off_table;          /* Zeichenabstandstabelle          */
```

```
        unsigned int    *dat_table;             /* Zeichensatzbild             */
        unsigned int    form_width;             /* Breite des Zeichensatzbilds */
        unsigned int    form_height;            /* Höhe des Zeichensatzbilds   */
        struct font_hdr *next_font;             /* Nächster Fontheader         */
} FONT_HDR;
```

Bedeutung von name:

name[0..15]: Zeichensatzname
name[16..31]: Attribute des Zeichensatzes (z.B. kursiv oder schattiert)

Bedeutung von lighten:

Lighten wird mit dem auszugegeben Zeichen per „AND" verknüpft und für die nächste Zeichenzeile um eine Bitposition rotiert. Es sind auch andere Masken als $5555 möglich, wobei sich teilweise interessante Effekte ergeben.

Bedeutung von skew:

Die Variable skew ist eine Maske für die Kursivausgabe von Zeichen. Ist in skew ein einer Zeichenzeile zugeordnetes Bit gesetzt, so wird diese Zeichenzeile um ein Pixel nach rechts verschoben ausgegeben. Andernfalls wird die Zeichenzeile nicht verschoben.
Das ATARI-VDI hat oftmals bei der Ausgabe von Proportionalfonts Probleme mit einer anderen Belegung als $5555. Das NVDI hat keine Probleme mit veränderten Masken wie z.B. $ffff oder $1111, so daß Neigungen zwischen 0-45 Grad mit skew eingestellt werden können.

Bedeutung von hor_table:

Es handelt sich hierbei um einen optionalen (daher auch das Flag) Zeiger auf eine Tabelle mit zwei Byte-Werten pro Zeichen, die bei vqt_width als left_delta und right_delta zurückgeliefert werden.

Begriffserklärungen

ASCII: Hinter diesen fünf Buchstaben verbirgt sich der „American Standard Code of Information Interchange". Dieser Code ordnet jedem Zeichen einen ASCII-Wert zu. ASCII diente z.T. als Grundlage für den IBM-Zeichensatz und somit auch für den ATARI-Zeichensatz.

binär: Ein Zahlensystem mit der Basis 2 wird als binäres Zahlensystem bezeichnet. Binären Zahlen ist in der Anleitung (z.B. bei den Linientypen) ein % vorangestellt.

Bit: Ein Bit kann nur zwei Werte annehmen, 0 oder 1. Das Bit bildet die Grundlage des binären Zahlensystems.

BUG: Zu deutsch „Wanze". Programmierer-Slang für einen Fehler.

Dispatcher: Zu deutsch in etwa „Verteiler", „Versender". Er verteilt die VDI-Aufrufe an eine entsprechende Unterroutine, die dann die Funktion ausführt.

Fast-RAM: Modelle des ATARI TT mit mehr als 4 Mbyte Speicher enthalten ein speziell organisiertes RAM, das zwar einigen Einschränkungen unterliegt (es kann z.B. nicht zur Bildschirmdarstellung verwendet werden) aber dafür den Vorteil einer wesentlich höheren Geschwindigkeit aufweist (ungefähr Faktor 2). Eine Verlagerung von Programmen in dieses spezielle RAM führt zu einer Geschwindigkeitssteigerung von 10-50%.

Font: Zeichensatz. GDOS verwendet derzeit bitmusterorientierte Zeichensätze. Im Gegensatz zu Vektorzeichensätzen sehen diese Zeichen bei starker Vergrößerung sehr unschön aus, haben aber den Vorteil wenig(er) Berechnungszeit und Speicher (Font-Caching) zu benötigen.

Gerätetreiber: Ein Programm, das vom GDOS geladen wird. Daher darf dieses Programm sich weder resident machen noch sich selbst mit einer GEMDOS-Funktion beenden. Der Anfang des Gerätetreibers wird vom GDOS angesprungen. Dort sollte ein Dispatcher die VDI-Aufrufe bearbeiten. Da so ein Gerätetreiber alle VDI-Funktionen beinhalten soll(te), ist das 4Programmieren eines Gerätetreibers mit einiger Arbeit verbunden. Gerätetreiber, die aus Teilen der INDEP.LIB von ATARI erstellt wurden, brillieren meist durch hohen Speicherverbrauch und lange Ausgabezeiten.

GEM: GEM ist der Sammelbegriff für AES („Application Enviroment Services") und VDI („Virtual Device Interface").

GEMDOS: GEMDOS ist (stark vereinfacht!) der für Disketten/ Harddisks und die Speicherwaltung zuständige Teil des Betriebssystems. GEMDOS ist ein MS-DOS-Clone, was wenigstens den Vorteil einer leichten Austauschbarkeit der Disketten mit sich bringt.

Handle: Kennung, die beim Öffnen einer Workstation zurückgegeben wird und für alle weiteren VDI-Aufrufe verwendet werden muß.

hexadezimal: Eine (falsche) Bezeichnung für ein Zahlensystem mit der Basis 16. Hexadezimalzahlen ist in der Anleitung ein $ vorangestellt.

M680xx: Eine Mikroprozessorserie der Firma Motorola Inc. Der ATARI ST enthält das Basismodell, den 16/32-Bitter M68000. Auf diversen Beschleunigerkarten finden Sie auch leistungsfähigere Prozessoren dieser Serie (M68020/M68030). Die einzig brauchbare Prozessorserie für „REAL PROGRAMMERS".

RAM: Abkürzung für 'Random Access Memory'. Gemeint ist der Hauptspeicher Ihres Systems.

Register: Mikroprozessoren besitzen eine begrenzte Anzahl eigener Speicherplätze, die sogenannten Register. Der Unterschied zwischen Registern und normalem Hauptspeicher liegt hauptsächlich darin, daß auf sie schneller zugegriffen werden kann.

ROM: ROM ('Read Only Memory') ist die Bezeichnung für einen Festwertspeicher. Er ist auslesbar, jedoch nicht beschreibbar. Gewöhnlich finden Sie das Betriebssytem in den ROMs.

TOS: Abkürzung für 'The Operating System'. Das Betriebssystem Ihres Rechners.

Treiber: s. Gerätetreiber.

:-) Keine Schwäche bei der Zeichensetzung, sondern ein Smiley (legen Sie mal den Kopf auf die linke Schulter!).

Funktionsumfang des NVDI-Bildschirmtreibers

Funktionsname		Bemerkungen
-1 v_set_app_buff		GEM/3-Funktion
1 v_opnwk		
2 v_clswk		
3 v_clrwk		
4 v_updwk		
5 v_escape		
	1 vq_chcells	
	2 v_exit_cur	
	3 v_enter_cur	
	4 v_curup	
	5 v_curdown	
	6 v_curright	
	7 v_curleft	
	8 v_curhome	
	9 v_eeos	
	10 v_eeol	
	11 v_curaddress	
	12 v_curtext	
	13 v_rvon	
	14 v_rvoff	
	15 vq_curaddress	
	16 vq_tabstatus	
	17 v_hardcopy	Systemroutine [Xbios(36)]
	18 v_dspcur	
	19 v_rmcur	
	61 v_sound	GEM 2.x-Funktion
	62 vs_mute	GEM 2.x-Funktion
	99 v_bez_qual	GEM/3-Funktion
	101 v_offset	Nicht von DR dokumentiert!
	102 v_fontinit	Vorsicht, nicht von DR dokumentiert!
6 v_pline		
6 v_bez		GEM/3-Funktion
7 v_pmarker		
8 v_gtext		
9 v_fillarea		
9 v_bez_fill		GEM/3-Funktion
11 v_gdp		
	1 v_bar	
	2 v_arc	
	3 v_pieslice	
	4 v_circle	
	5 v_ellipse	

Funktionsname	Bemerkungen
6 v_ellarc	
7 v_ellpie	
8 v_rbox	
9 v_rfbox	
10 v_justified	
13 v_bez_on GEM/3-Funktion	
13 v_bez_off GEM/3-Funktion	
12 vst_height	
13 vst_rotation	
14 vs_color	
15 vsl_type	
16 vsl_width	
17 vsl_color	
18 vsm_type	
19 vsm_height	
20 vsm_color	
21 vst_font	
22 vst_color	
23 vsf_interior	
24 vsf_style	
25 vsf_color	
26 vq_color	
28 v_locator	
30 v_choice	
31 v_string	
32 vswr_mode	
33 vsin_mode	
35 vql_attributes	
36 vqm_attributes	
37 vqf_attributes	
38 vqt_attributes	
39 vst_alignment	
100 v_opnvwk	
101 v_clsvwk	
102 vq_extnd	Entsprechend GEM 2.x erweitert
103 v_contourfill	
104 vsf_perimeter	
105 v_get_pixel	
106 vst_effects	
107 vst_point	
108 vsl_ends	
109 vro_cpyfm	
110 vr_trnfm	
111 vsc_form	
112 vsf_udpat	

Funktionsname	Bemerkungen
113 vsl_udsty	
114 vr_recfl	
115 vqin_mode	
116 vqt_extent	
117 vqt_width	
118 vex_timv	
119 vst_load_fonts	
120 vst_unload_fonts	
121 vrt_cpyfm	
122 v_show_c	
123 v_hide_c	
124 vq_mouse	
125 vex_butv	
126 vex_motv	
127 vex_curv	
128 vq_key_s	
129 vs_clip	
130 vqt_name	
131 vqt_fontinfo	
132 vqt_justified	GEM 2.x-Funktion

Index der VDI-Funktionen





ATARI ST ATARI TT



